

NUMERICAL SIMULATION OF VISCOUS ACCRETION DISKS

By

RONALD EUGENE DRIMMEL

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1995

For my parents,
for the poor, and to
Sacred Truth and Beauty.

ACKNOWLEDGMENTS

First, I must thank my advisor, Dr. James Hunter, Jr., who has always had high expectations of me. He has not only given me guidance, but has taught me by example many of the important qualities that a theorist should possess, including the qualities of being careful, conservative, hard working, and honest. I thank him as well for his conversation, and attempts to pass on to me some of his experience and physical intuition. I also thank him as a teacher, especially for the lesson that one should take the time to start from fundamental theory, and the importance of deriving even “basic” equations before applying them to a problem. After my time in graduate school, I am certainly the richer for our association. I must also thank the other members of my committee, Drs. H. Kandrup, H. Smith, S. Dermott, and J. Klauder, for their helpful discussions, questions, and recommendations.

I’d like to thank Chad Davies, officemate extraordinaire, who offered encouragement both as a fellow scientist and friend. I must also thank Clayton Heller for his helpful input and questions on numerical aspects of this work, as well as Nikos Hiotelis, who made available to the Astronomy Department the original N-body code, TREECOD, and assisted me in getting started. To the many other fellow astronomers and students, whose conversation I have also benefited from, I give thanks. This work was also made possible by a NASA GSRP Fellowship, which gave me generous support during most of this research, which also was supported in part by the University of Florida and the IBM Corp., through their Research Computing Initiative at the Northeast Regional Data Center of Florida, without which the numerical work would not have been possible.

On a more personal and broader note, I would like to thank Jonathan Potter for his friendship, and keeping me laughing when I needed it most, as well as all my other friends, who have assisted me more than they will know. This includes Orlando Espin, for teaching me that my love for astronomy and my love for the poor were not in contradiction, but that I can serve all God's children with and through my work as an astronomer. Also on a spiritual note, and another friend who needs mention, is Mary Flanagan, who gently guided me on my way. I have been blessed as well by the entire community of Saint Augustine Catholic Church, which has provided sustaining strength, healing, and abundant joy. I give gracious thanks as well to Teilhard de Chardin, for helping me see the big picture. To my parents, who have always believed in me, I am most grateful. Lastly, I must give ultimate thanks to my God, Master of the Universe and Lover of Souls, who created all in beauty, and who blessed me with the eyes to see it.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	iii
LIST OF TABLES	vii
LIST OF FIGURES	viii
ABSTRACT	xi
CHAPTERS	
1. INTRODUCTION	1
2. METHODS	6
Development of the Hydrodynamic Code	6
N-body Seed Code	6
SPH	10
Variable Smoothing Length and Neighbor Searching	14
Hydrodynamic Equations	16
Integration Method	19
Gravitational Softening	20
Accretion	23
Summary of Code Parameters	25
Tests	26
Maclaurin Disk	26
Two-Dimensional Shocks	27
Analysis of Disks	30
Evaluation of Effective Shear Viscosity	30
Evaluation of Modes and Their Frequencies	33
3. INITIALIZATION OF DISKS	39
Two-dimensional Disks	40
Maclaurin Disk	40
Exponential Disks	43
Inner Disks	51
4. SIMULATION OF TWO-DIMENSIONAL DISKS	56
Parameters	56
Evolution of Disks	60
General Features	60
Modal Evolution	62
Accretion	65
Tests of Code Parameters	71

5. FORMATION OF SATELLITES	84
Orbital Evolution	85
Formation Conditions	90
Encounter Model	95
6. SUMMARY AND CONCLUSIONS	108
Results	108
Code Development	108
Modal Evolution	111
Accretion	112
Formation of Satellites	113
Future Work	115
A.T2DSPH	119
B.Algorithm for T2DSPH	262
C.PROGRAM EXPD	263
BIBLIOGRAPHY	277
BIOGRAPHICAL SKETCH	281

LIST OF TABLES

Table 2–1: Code Parameters	25
Table 4–1: Models	58
Table 4–2: Models at $T=28$ dynamical times	61
Table 4–3: Accretion Rates ($\times 10^{-3}$)	67
Table 4–4: Normalized density amplitude growth rates and saturation levels.	69
Table 5–1: Satellites	85

LIST OF FIGURES

Figure 2–1: Illustration of a two-dimensional hierarchical tree cell structure with 17 particles. There are two levels of subcells below the root cell.	7
Figure 2–2: Surface density and velocity profile of the Maclaurin disk after 26 dynamical times.	27
Figure 2–3: Surface density of three two-dimensional accretion shocks. The bottom accretion shock’s surface density is offset from the top by -2 , and the middle by -1 , from the top accretion shock.	29
Figure 2–4: Normalized power spectrum of modes of model A2 (see Chapter 4) at Time = 21 dynamical times.	35
Figure 2–5: Dynamic power spectrum of the normalized density for mode = 2 of model A2.	36
Figure 2–6: Maximum of the power spectrum, in modes = 1 – 4, of the normalized density in model A2.	37
Figure 2–7: The average Lombe periodogram of normalized density temporal fluctuations along Cartesian axes for model A2.	38
Figure 3–1: Initial distribution of particles for the Maclaurin Disk.	41
Figure 3–2: The initial Toomre Q parameter (solid line) and tangential velocity (dotted line) with respect to radius for a $M_c/M_D = 3$ disk.	46
Figure 3–3: Initial distribution of particles in an exponential disk.	48
Figure 3–4: Initial density profile of an exponential disk with $M_D = 0.25$ and a scale length of $r_s = 0.25$. Each point corresponds to the estimated density at each particle position, evaluated with the SPH method, while the solid line is the intended density profile.	49
Figure 4–1: Particle distribution of $M_c/M_D = 3$ models at Time = 10.0 dynamical times.	73

Figure 4-2: Particle distribution of $M_c/M_D = 3$ models at Time = 29.5 dynamical times.	74
Figure 4-3: Particle distribution of $M_c/M_D = 1$ models at Time = 10.0 dynamical times.	75
Figure 4-4: Particle distribution of $M_c/M_D = 1$ models at Time = 30.0 dynamical times.	76
Figure 4-5: Maximum power in modes 1 through 4 for model B2.	77
Figure 4-6: Resonances in an accretion disk with $M_c/M_D = 3$. Solid lines correspond to the corotation resonance, the dotted lines to the Inner Lindblad resonance, and the dashed lines to the Outer Lindblad resonance. . . .	78
Figure 4-7: Mass accretion, as a fraction of initial disk mass, for models A1 through A3.	79
Figure 4-8: Mass accretion, as a fraction of initial disk mass, for B models.	79
Figure 4-9: Mass accretion, as a fraction of initial disk mass, for C models.	80
Figure 4-10: Mass accretion, as a fraction of initial disk mass, for D models.	80
Figure 4-11: Mass accretion in models A2, A4, and A5, as a fraction of the initial disk mass.	81
Figure 4-12: Constant mass accretion rates for $M_c/M_D = 3$ accretion disks.	82
Figure 4-13: Constant mass accretion rates for $M_c/M_D = 1$ accretion disks.	82
Figure 4-14: Mass accretion of the encounter model (solid line) compared to the mass accretion of model D2 (dotted line).	83
Figure 4-15: Mass accretion, as a fraction of initial disk mass, for model D2 is shown with three test models.	83
Figure 5-1: Evolutionary sequence of model D2 showing formation of satellite. . . .	96

Figure 5-2: Evolutionary sequence of model D2 continued.	97
Figure 5-3: Final configuration of model D2, showing the position of the satellite at earlier times.	98
Figure 5-4: Radial density profile and particle distribution of satellite D2-1.	99
Figure 5-5: Final configuration of model B2, showing the position of the satellites at earlier times.	100
Figure 5-6: Final configuration of model B3, showing the position of the satellites at earlier times.	101
Figure 5-7: Final configuration of model B1, showing the position of the satellites at earlier times.	102
Figure 5-8: Final configuration of model D1, showing the positions of the satellites at earlier times. Satellite 4, which has been reabsorbed by the disk, is not shown.	103
Figure 5-9: Encounter model.	104
Figure 5-10: Encounter model continued.	105
Figure 5-11: Encounter model continued.	106
Figure 5-12: Encounter model at Time = 16.0. Note the three satellites that have formed in the tidal tail.	107

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

NUMERICAL SIMULATION OF VISCOUS ACCRETION DISKS

By

RONALD EUGENE DRIMMEL

August, 1995

Chairman: James H. Hunter, Jr.

Major Department: Astronomy

A two- and three-dimensional numerical hydrodynamics FORTRAN code is developed to model viscous accretion disks, employing the method of smoothed particle hydrodynamics. The effective shear viscosity present in the code is evaluated. Using a polytropic equation of state, models of self-gravitating accretion disks are evolved with central mass to disk ratios of 1 and 3, with ratios of specific heats of 2 and $5/3$, and with an artificial viscosity parameter of 1, 0.5, 0.25. From these models it is found that a characteristic mass accretion rate, constant with time, is maintained by the accretion disks, and that mass accretion is *inversely* proportional to the strength of the local shear viscous force. This is a consequence of the effect of local viscous forces upon the global non-axisymmetric modes that primarily drive accretion in these disks. In addition, the formation of satellites is observed in models which also develop a dominating $m=1$ mode. The relevance of these satellites to planet formation is speculated.

CHAPTER 1 INTRODUCTION

Accretion disks form an important class of astrophysical objects, occurring on many scales in a variety of contexts. In quasars and AGNs they are believed to be the central driving engine responsible for the extreme luminosities generated by these objects. In a more common manifestation, they are believed to be a natural product of the star formation process (Shu *et al.*, 1987), and to be the progenitors of potential planetary systems. As a structure that preprocesses infalling material before it accretes onto the central protostar, the accretion disk is referred to as a protostellar disk. Later, when mass accretion has virtually ended and the disk mass is small compared to the young star at its center, it is designated as a protoplanetary disk. This work, though it addresses accretion disks in general, is focused on accretion disks with the general characteristics of massive protostellar disks: the self-gravity of the disk will be important, and the central object, representing the protostar, will be small compared with the spatial dimensions of the disk.

Though the existence of protostellar disks was first theoretically argued as a natural consequence of the conservation of angular momentum, the observational evidence for these objects has greatly improved. Initially the observational evidence was indirect. Infrared (IR) excesses were observed around young stellar objects, particularly T Tauri stars, though the star itself was unobscured by the material radiating at lower temperatures (Rydgren & Zak, 1987). Ultraviolet (UV) emission lines are associated with these objects as well. Other indirect evidence is the angular momentum regulation of T Tauri stars

with IR excesses (Edwards *et al.*, 1993). The most plausible, and simple, of explanations for these observations was that the material responsible for the excess IR emissions was in a flat, disk-like structure, and that material accreting from the disk onto the star was producing the UV emission. Many of these inferred disks have little mass, and are still thought to be examples of disks at a later stage in evolution.

More massive disks were inferred from young stellar objects with much larger IR excesses (Hillenbrand *et al.*, 1992). The “flat-top” IR spectra of these objects were initially explained by massive disks that were generating their own luminosity through viscous processes (Adams *et al.*, 1988). However, more recently the spectra of these objects were shown as more probably resulting from a dusty, infalling envelope (Whitney & Hartmann, 1993; Hartmann, 1993). Hence, to this point, the indirect evidence for large gaseous disks around young stars has provided enticing, but not decisive, evidence. However, with new and improved instrumentation, such as the JCMT-CSO submillimeter interferometer and the serviced Hubble Space Telescope, direct evidence of gaseous disks around young stellar objects has been observed. (Koerner *et al.*, 1993; Lay *et al.*, 1993)

Numerically many have shown, as per expectations (Lin & Pringle, 1990), that massive disks do form from a collapsing cloud possessing angular momentum equivalent to that observed in molecular cloud cores (Bodenheimer *et al.*, 1990; York *et al.*, 1993; York *et al.*, 1995). In such a self-gravitating disk, nonaxisymmetric modes play a crucial role, as they effectively transport angular momentum, thereby driving the evolution of the disk. In addition, ubiquitous shear viscous forces are an important transport mechanism in a gaseous disk; nonaxisymmetric modes may dominate when present, but viscous forces

will always be present to some degree. In general the resulting angular momentum transport will be outwards, causing the disk to become more extended, while the bulk of the mass flows inward, resulting in accretion onto the central object (Lynden-Bell & Pringle, 1974). The strength of the viscous force has been inferred from the lifetimes of the disks themselves, and is often termed as anomalous viscosity, as the strength of the viscous forces are much stronger than can be explained by the kinetic molecular viscosity of the gas itself. The most common type of viscosity that has been suggested is turbulent viscosity, though several mechanisms for inducing the turbulence have been suggested. These mechanisms include convection induced by thermal gradients, which requires an optically thick disk, and a magneto-rotational instability, which requires the presence of a weak magnetic field (Balbus & Hawley, 1991; Hawley & Balbus, 1991). In this work I do not attempt to shed light on the mechanism responsible for the turbulence, but will nonetheless assume that turbulence is responsible for the shear viscosity in the accretion disk. These two mechanisms of angular momentum transfer, the global nonaxisymmetric modes and the local viscous processes, do not act independently, but affect one another.

The theory of accretion disks, motivated in large degree by their occurrence in some binary systems, particularly cataclysmic variables, has been most fully developed for the thin Keplerian disk (Pringle, 1981). In this regime the self-gravity of the disk is neglected, and only pressure supports the vertical structure. In contrast, the theory of self-gravitating accretion disks is less developed. This is in part due to the nonlinearity of the problem, which dictates that numerical studies are necessary to understand these disks. Early numerical work on self-gravitating accretion disks began with N-body

modeling (Cassen *et al.* 1981; Tomley *et al.*, 1991). Papaloizou and Savonije (1991) analytically investigate the instabilities that exist in two-dimensional self-gravitating disks, and numerically follow the evolution of these unstable modes. Their numerical method was to solve the hydrodynamic equations on a polar grid, which necessitated the unphysical assumption of rigid inner and outer boundary conditions. They confirm the importance of the nonaxisymmetric modes in redistributing the mass of the bounded disk, though the subsequent evolution they observe characterizes real disks only in a broad sense. In addition, their numerical code has an indeterminate amount of shear viscosity, which also contributes to the angular momentum transport and mass redistribution that they observe in their models.

Only recently have self-gravitating three-dimensional disks been investigated through numerical simulation. The stability of three-dimensional tori has been investigated both analytically and numerically (Papaloizou & Pringle, 1984; Papaloizou & Pringle, 1985; Zurek & Benz, 1986; Tohline & Woodward, 1992). Most recently Laughlin and Bodenheimer (1994) have investigated the initial evolution of a disk that is formed from a previous calculation of a collapsing gas cloud. In this later work ring modes that had formed in a two dimensional collapse calculation were shown to be unstable in a three-dimensional hydrodynamic simulation.

In this work I extend the numerical work of Papaloizou and Savonije (1991) by providing a model that more closely resembles that of an astrophysical accretion disk, but that is also restricted to two dimensions. In particular, the hydrodynamic evolution of the disk is followed with a smoothed particle hydrodynamics (SPH) code, which

approximates a gas with a finite number of particles (Benz, 1990; Monaghan, 1992). The evolution in phase space of these particles is determined by the Lagrangian form of the hydrodynamic equations. As this method is not restricted to a fixed grid with rigid boundaries, the disk is allowed to expand naturally in radius, and an inner boundary condition allows accretion onto the central gravitating object. For the sake of continuity with Papaloizou and Savonije (1991), the disks are initially given an exponential density profile, though unlike them the disks are perturbed by introducing noise in the density profile. In this way all unstable modes will be excited. The primary disk parameters which are varied are the star to disk mass ratio, the ratio of specific heats, and the effective shear viscosity of the disks. Like Papaloizou and Savonije (1991), a polytropic equation of state is used to describe the gas. In particular, the effect of the shear viscosity on the evolution of self-gravitating disks is investigated.

The following chapter describes the numerical method employed to model accretion disks and its implementation in a FORTRAN code named T2DSPH. A listing of the code itself is provided in Appendix 1. Tests of the code are also described, including an evaluation of the effective shear viscosity that is present. Chapter 3 gives the details of the initialization of the models that are evolved, an important and nontrivial aspect of SPH. In Chapter 4 the initial parameters of the accretion disk models are first briefly described, followed by a presentation of the results of the modeling. The evolution of the nonaxisymmetric modes and the mass accretion observed in the models are discussed. Chapter 5 addresses the formation of satellites that occur in some of the models, and Chapter 6 summarizes the results and speculates on their significance.

CHAPTER 2 METHODS

Development of the Hydrodynamic Code

In this chapter the development of the numerical hydrodynamic code, named T2DSPH, is described. A listing of the code is given in Appendix A, and an algorithm of the code is given in Appendix B. Test simulations are also described, including the evaluation of the effective shear viscosity present in the method. In addition, methods for analysing the disks are discussed in the last section.

N-body Seed Code

The code developed to evolve accretion disks is itself evolved from a version of Hernquist's N-body code called TREECOD (1987). This code uses a method of calculating the gravitational forces on a system of particles, due to the collective influence of the entire system of particles, using a method termed the hierarchical tree method. At any particular time a system of N particles is spatially described, in Cartesian coordinates, by a set of cells hierarchically organized. The root cell encompasses the entire system; then succeeding levels of subcells are created. In two dimensions each cell will have four subcells, while in three dimensions each cell has eight subcells. This is accomplished by simply dividing a cell in half in each dimension (see Figure 2-1). The number of particles in each cell is found, and successive levels of cells are created until subcells have either one or no particles in it. Cells with a single particle can be considered the

leaves of the tree. The spatial aspect of the tree is expressed by allocating to memory the position and dimension of each cell. Hence, the tree structure completely describes the spatial structure of a system of particles. Further, by recording for each cell the pointer to its parent cell, this tree may be ascended or descended.

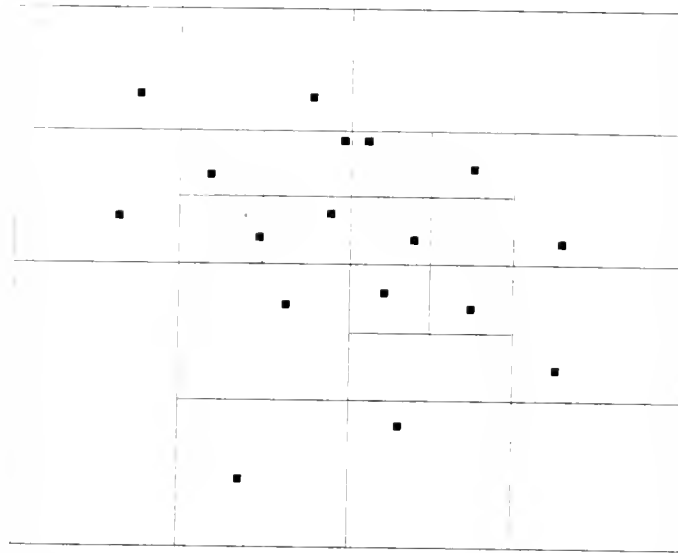


Figure 2-1: Illustration of a two-dimensional hierarchical tree cell structure with 17 particles. There are two levels of subcells below the root cell.

Moving from leaves to root, the mass and the center of mass of each cell is quickly determined; in describing the spatial distribution of a system of N massive particles, the mass distribution of the system is also described. This hierarchical tree structure can now be utilized to calculate the gravitational forces on any single particle, and allows the implementation of an advantageous approximation: the number of gravitation terms is made significantly less than the $N-1$ terms a direct sum over all the other particles would yield, by treating cells as gravitating particles. In other words, a single gravitational term due to a given cell, its mass taken to be at the position of its center of mass, is used to

represent the collective gravitational influence of all the particles within the cell. Using this approach, the gravitational force of a system of N particles on one of its members can be approximated with a sum of N_t terms due to a set of N_t cells, where $N_t \ll N$.

The choice of cells used in the gravitational force calculation will in general be different for each particle, and the choice of which cells to use will determine the accuracy of the approximation. To determine the set of cells used for a given particle, the tree is descended from root to leaves, with the descent continuing to the next level of subcells until the cell subtends an angle, as “viewed” from at the particle, which is less than a specified tolerance angle, θ . In effect this method will treat the influence of nearby neighbors as a direct sum, but simplify the influence of particles at a distance by grouping them into larger cells. This approximation is then based on the philosophy that the local gravitational field is not sensitive to the detailed spatial distribution of particles at a distance. The accuracy of the approximation is increased further by including higher order terms in a multipole expansion of the mass distribution within the cells. Taking the mass to be at the center of mass, as described above, we already have the monopole expansion term. The next higher order correction term is the quadropole term; since the center of mass is used as the expansion center the dipole term is zero.

Hernquist has found that for a system with 4096 particles that a tolerance angle of $\theta < 0.8$ radians gives a relative error in the i th component of the acceleration, $A(\delta a_i)/\bar{a}_i$, of less than 1%, where he defines the mean absolute deviation from the mean error,

$$A(\delta a_i) = \frac{1}{N} \sum_1^N |\delta a_{ij} - \bar{\delta a}_i|, \quad (2.1)$$

and the absolute average acceleration,

$$\bar{a}_i = \frac{1}{N} \sum_1^N \left| a_{ij}^{\text{direct}} \right|. \quad (2.2)$$

He define the mean error as

$$\bar{\delta a}_i = \frac{1}{N} \sum_1^N \delta a_{ij}, \quad (2.3)$$

where $\delta a_{ij} = a_{ij}^{\text{tree}} - a_{ij}^{\text{direct}}$ is the difference, in the i th component of the acceleration, as calculated by the tree hierarchical and direct summation methods for particle j . This was measured by Hernquist (1987) for a Plummer sphere, which has a density profile of

$$\rho(r) = \frac{3M}{4\pi} \frac{r_0^2}{(r^2 + r_0^2)^{5/2}}, \quad (2.4)$$

where M is the mass of the system and r_0 is the scale length. For a typical two-dimensional exponential disk, I measured an average relative error in the accelerations, defined as

$$\frac{1}{N} \sum_{j=1}^N \left| (\delta a_{ij} - \bar{\delta a}_i) / a_{ij}^{\text{direct}} \right|. \quad (2.5)$$

For both an axisymmetric and a nonaxisymmetric disk, with a tolerance angle $\theta = 0.8$, I found the average relative error to be of the order of 1%.

The computational time required to sample the gravitational force field of an N -body system at N points is proportional to the number of gravitational terms to be calculated. Therefore the computational time of the direct sum method is of the order of N^2 , whereas Hernquist finds that the tree hierarchical system is of the order of $M \log(N)$. Models of accretion disks presented in this work employ approximately 5000 to 10000 particles;

for these numbers of particles the increase in efficiency, as compared to using a direct summation method, is over a thousand fold. Nevertheless, there are other methods for estimating the gravitational field of an N-body system that are more computationally efficient, such as grid methods that use fast Fourier techniques to evaluate the gravitational field on a set of grid points. The gravitational force upon a single particle is then interpolated from the nearest points. While such methods are faster, the tree hierarchical method possesses several attractive advantages. Grid methods are limited in resolution to the grid size, whereas the tree method's local resolution is limited by the local particle distribution itself, or the gravitational softening parameter, since its approximation is only applied to particles at a distance. In addition, tree methods are not limited to the size or geometry of a grid, as the root cell is resized at every step and empty cells do not take up memory. Hence tree hierarchical methods conserve mass exactly. These characteristics make tree methods ideal for modeling violent encounters, as well as models that have a broad range in particle densities.

SPH

Using this N-body code as a skeleton, a two-dimensional hydrodynamic code was developed, called T2DSPH (Appendix A). The numerical hydrodynamic method chosen to model accretion disks is known as smoothed particle hydrodynamics (SPH), and in developing the code I followed much of the guidance given by Hernquist and Katz's own combination of SPH and the tree hierarchical method (1989). A three-dimensional version of this hydrodynamics code was then developed and dubbed T3DSPH.

In essence, “SPH is an interpolation method that allows any [continuous] function to be expressed in terms of its value at a set of disordered points – the particles,” (Monaghan 1992). A finite number of particles is used to describe the density and bulk velocity field of a fluid; hydrodynamical quantities needed to calculate the acceleration at the points are estimated; and the accelerations are then applied to the particles using an appropriate integrator. SPH, then, is a Lagrangian technique of solving the hydrodynamical equations, employing a finite number of particles to approximate the fluid. A great advantage of such a method is that it is quite general, being easily amendable to any choice of equation of state. Also, a Lagrangian approach has an advantage over an Eulerian technique, which must employ a grid, in that no symmetry is imposed or assumed, and further, the simulation is not confined to a box of finite size. Reviews of this technique, which has been popular in recent years to model a variety of astrophysical problems, are provided by Monaghan (1992) and Benz (1990). The following discussion of the mathematical basis of SPH follows the discussion given both in Benz (1990), and Hernquist and Katz (1989).

In SPH, quantities are estimated using the smoothed estimate,

$$\langle f(\mathbf{r}) \rangle = \int f(\mathbf{r}') W(\mathbf{r} - \mathbf{r}', h) d\mathbf{r}', \quad (2.6)$$

where W is a smoothing kernel and h a smoothing length. The kernel W is spherically symmetric and normalized:

$$\int W(\mathbf{r}, h) d\mathbf{r} = 1. \quad (2.7)$$

If the kernel W is a function strongly peaked about $\mathbf{r} = 0$ then the estimate $\langle f \rangle$ can be

written as a Taylor expansion:

$$\langle f(\mathbf{r}) \rangle = f(\mathbf{r}) + c(\nabla^2 f)h^2 + O(h^3), \quad (2.8)$$

where the coefficient $c = \int u^2 h^3 W(u) du$, ($u = |\mathbf{r} - \mathbf{r}'|/h$), is independent of h . The term proportional to h has vanished because W is an even function in \mathbf{r} . Hence it is said that the smoothed estimate of the function f is second order accurate in h . If the quantity f is known on a finite set of points described by a point distribution,

$$n(\mathbf{r}) = \sum_{j=1}^N \delta(\mathbf{r} - \mathbf{r}_j), \quad (2.9)$$

then the integral expression for the estimate $\langle f \rangle$ (equation 2.3) can be written as a sum:

$$\langle f(\mathbf{r}_i) \rangle = \sum_j^N \frac{m_j f(\mathbf{r}_j)}{\rho(\mathbf{r}_j)} W_{ij}, \quad (2.10)$$

using the relation

$$\left\langle \frac{A(\mathbf{r})}{B(\mathbf{r})} \right\rangle = \frac{\langle A(\mathbf{r}) \rangle}{\langle B(\mathbf{r}) \rangle} + O(h^2). \quad (2.11)$$

I have introduced the abbreviation $W_{ij} = W(|\mathbf{r}_i - \mathbf{r}_j|, h)$, and written $\langle n_j \rangle$ in the form $\rho(\mathbf{r}_j)/m_j$, where m_j is the particle mass and $\rho(\mathbf{r}_j)$ is the density at the particle j . Using equation (2.7), the density is estimated using

$$\rho(\mathbf{r}_j) = \sum_j^N m_j W_{ij}. \quad (2.12)$$

Because of the form of the estimate [equation (2.7)], it is most convenient to estimate quantities with the general form ρA :

$$\langle \rho A \rangle_i = \sum_j^N m_j A_j W_{ij}. \quad (2.13)$$

This convention is considered a “golden rule of SPH” implementation by Monaghan (1992).

Various smoothing kernels can be used; in this work the spherically symmetric kernel

$$W(r, h) = \frac{1}{8\pi h^3} \begin{cases} 1 - 6u^2 + 6u^3, & \text{if } 0 \leq u \leq 1/2 \\ 2(1 - u)^3, & \text{if } 1/2 \leq u \leq 1 \\ 0, & \text{otherwise,} \end{cases} \quad (2.14)$$

is used, where $u = r/h$. This kernel is a fourth order basis spline, normalized for three dimensions. In two dimensions normalization yields the leading factor of $10/7\pi h^2$. I should also note that the traditional way of writing the above kernel is with $h \rightarrow 2h$, so that W is nonzero for $r < 2h$ rather than h , but the above form is more convenient numerically. The kernel above is the one most widely used in the SPH community, and has the advantage that it defines a local interpolation since, being nonzero only when $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j| < h$, neighboring particles alone are used to make the estimate. Higher order basis splines were tested but did not yield significantly better estimates of a given function for equivalent smoothing lengths, and gave worse results when derivatives were estimated.

One of the powerful features of SPH is the ease with which the gradients of quantities can be estimated:

$$\langle \nabla f_i \rangle = \sum_j \frac{m_j f_j}{\rho_j} \nabla W_{ij}. \quad (2.15)$$

(For the sake of brevity I will in general be writing f_i for $f(\mathbf{r}_i)$, and sums are to be understood as being over neighboring particles only.) Note that the kernel is symmetric ($W_{ij} = W_{ji}$), while its gradient is antisymmetric ($\nabla_{\mathbf{r}_i} W_{ij} = -\nabla_{\mathbf{r}_j} W_{ji}$). The divergence of a vector quantity, such as velocity, can take a similar form. However, if the estimate

of a divergence is to be used in an equation of motion, it must be written in a form that is antisymmetric in i and j in order to conserve angular momentum when particles interact. Thus, for the quantity $\nabla \cdot \mathbf{v}$, the identity $\rho \nabla \cdot \mathbf{v} = \nabla \cdot \rho \mathbf{v} - \mathbf{v} \cdot \nabla \rho$ is used to derive the following estimate:

$$\langle \nabla \cdot \mathbf{v} \rangle_i = -\frac{1}{\rho_i} \sum_j m_j \mathbf{v}_{ij} \cdot \nabla W_{ij}, \quad (2.16)$$

where $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$ (see Monaghan 1992).

Variable Smoothing Length and Neighbor Searching

So far I have given the SPH formalism in the case where the smoothing length h is a constant. In general, however, particles are given individual smoothing lengths h_i , which are adjusted, both in space and time, so that the SPH particles have approximately the same number of neighbors. This refinement will insure that quantities throughout a model are estimated with comparable accuracy, and enables SPH to adjust the local resolution as the local number density changes during the course of a model's evolution; denser regions will have finer resolution than diffuse regions. However, the form of the kernel W_{ij} , with its dependence on h , is now uncertain. Again, to insure that angular momentum will be conserved, the kernel is rendered symmetric in i and j by defining the smoothing length $h \equiv h_{ij} = (h_i + h_j)/2$. Neighboring particles j will now be used that satisfy the condition $r_{ij} < h_{ij}$. Another way to symmetrize the kernel is to define $W_{ij} = (W(r_{ij}, h_i) + W(r_{ij}, h_j))/2$, which is the form used by Hernquist and Katz (1989).

Determining the set of neighboring particles for each particle (those satisfying the condition $r_{ij} < h_{ij}$) is known as “neighbor searching”, and can be one of the more time intensive tasks in an SPH program. In the version of SPH developed for this work, the neighbor lists are efficiently compiled using the already existing hierarchical tree data structure used in the N-body portion of the code (see the beginning of this chapter). First, the list of neighbors within h_i is found by descending the tree, starting at the root, and continuing to sublevels of those cells that intersect the neighborhood of particle i , defined by h_i . When the descent reaches a particle it is stored in the neighbor list of particle i , provided it satisfies the condition $r_{ij} < h_{ij}$. At the same time, if $r_{ij} > h_j$ for neighbor j particle i is added to the list of neighbors of particle j , since these particles will not be found when the neighbor search is done for particle j .

The adjustment of the smoothing lengths at each time step is adapted from the procedure outlined by Steinmetz and Müller (1993), which uses a predicted smoothed density, ρ_i^* , to find a new smoothing length:

$$h_i = \zeta \left(\frac{\bar{m}}{\rho_i^*} \right)^{1/2}, \quad (2.17)$$

where ζ is set to 1.3 and \bar{m} is the average mass of the neighboring particles. The predicted smoothed density is calculated by using

$$\rho_i^* = \rho_i^{\text{smooth}} + \Delta t \left(\frac{d\rho}{dt} \right)_i^{\text{smooth}}, \quad (2.18)$$

where

$$\left(\frac{d\rho}{dt} \right)_i^{\text{smooth}} = \rho_i^{\text{smooth}} (\nabla \cdot \mathbf{v})_i - \frac{2}{\rho_i} (\rho^2 \nabla \cdot \mathbf{v})_i^{\text{smooth}} \quad (2.19)$$

and

$$(\rho^2 \nabla \cdot \mathbf{v})_i^{\text{smooth}} = \sum_j m_j \rho_j (\nabla \cdot \mathbf{v})_j W_{ij} . \quad (2.20)$$

All smoothed quantities, indicated by superscript, are estimated using the smoothing length $h_i^* = h_i/0.8$.

This method maintains the number of neighbors, N_s , between 16 and 24 on the average, except for the outer (diffuse) portions of the disks, where N_s can drop below 10. In order to give reliable estimates the interparticle distance between neighbors should be less than $0.5h_i$ for the kernel used [equation (2.10)]. This requires that $N_s \geq 13$ in two dimensions, and $N_s \geq 34$ in three dimensions. If there are large density contrasts in the outer regions, the adjustment technique may also cause N_s for a particle to jump to some large value. Therefore, as a practical matter, h_i is adjusted if N_s is greater than 60 or less than 10. As this adjustment takes place while the neighbor lists are being compiled, errors in the neighbor lists of nearby particles will be introduced, which must be corrected.

Hydrodynamic Equations

To evolve the fluid, represented by a finite number of particles, the acceleration and velocity of the fluid at each particle is applied to the particles themselves. The particles can then be thought of as being parcels of fluid, or alternatively, as representative particles moving with the fluid. In either case, such a system's dynamic evolution is described by the Lagrangian form of the hydrodynamic equations:

$$\begin{aligned} \frac{d\mathbf{r}_i}{dt} &= \mathbf{v}_i \\ \frac{d\mathbf{v}_i}{dt} &= -\frac{1}{\rho_i} \nabla P_i + \mathbf{a}_i^{\text{visc}} - \nabla \Phi_i, \end{aligned} \quad (2.21)$$

where P_i is the pressure, $\mathbf{a}_i^{\text{visc}}$ is the artificial viscosity term, and $\nabla \Phi_i$ is the acceleration due to the self gravity of the system, evaluated using the hierarchical tree method. The subscript i indicates that the above set of equations is applied to each particle. The acceleration on each particle is evaluated by estimating the terms on the right hand side of the equation of motion using the SPH formalism. Care must be taken that each term is in a form which is antisymmetric in i and j , so that the mutual forces of particles will be equal in magnitude, but opposite in direction. In conjunction with the forces being central, the antisymmetry of the force terms will insure that angular momentum is conserved. Following Hernquist and Katz (1989),

$$-\frac{1}{\rho_i} \nabla P_i + \mathbf{a}_i^{\text{visc}} = - \sum_j m_j \left(2 \frac{\sqrt{P_i P_j}}{\rho_i \rho_j} + \Pi_{ij} \right) \nabla W_{ij}, \quad (2.22)$$

where the artificial viscous term, taken from Monaghan (1992), is

$$\Pi_{ij} = \begin{cases} \frac{-\alpha c_{ij} \mu_{ij} + \beta \mu_{ij}^2}{\rho_{ij}}, & \mathbf{v}_{ij} \cdot \mathbf{r}_{ij} < 0 \\ \frac{-\alpha c_{ij} \mu_{ij}}{\rho_{ij}}, & \mathbf{v}_{ij} \cdot \mathbf{r}_{ij} > 0 \end{cases}, \quad (2.23)$$

where

$$\mu_{ij} = \frac{h \mathbf{v}_{ij} \cdot \mathbf{r}_{ij}}{r_{ij}^2 + \eta^2}. \quad (2.24)$$

In the above equations $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, and $c_{ij} = (c_i + c_j)/2$ is the average of the sound speed at i and j . The term $\eta^2 = 0.01 h^2$ is to prevent singularities. The viscosity parameters α and β control the strength of the artificial viscous term. An artificial viscous term is introduced into SPH to model the bulk viscosity necessary to reproduce shocks. Without the dissipation that this term effects, there would be an “interpenetration” of particles, the random kinetic energy would increase, and the particles would no longer describe the bulk motion of a fluid. However, it also introduces an effective shear

viscosity. To reduce the effective shear viscosity Benz (1990) has introduced a switch in the form of a multiplicative factor on μ_{ij} , ($\mu_{ij} \rightarrow \mu_{ij} f_{ij}$). The factor f_{ij} is the average of f_i and f_j , where

$$f_i = \frac{|\langle \nabla \cdot \mathbf{v} \rangle|_i}{|\langle \nabla \cdot \mathbf{v} \rangle|_i + |\langle \nabla \times \mathbf{v} \rangle|_i + 0.001 c_i / h_i}, \quad (2.25)$$

and the curl of the velocity field is given by the estimate

$$\langle \nabla \times \mathbf{v} \rangle_i = \frac{1}{\Sigma_i} \sum_j m_j \mathbf{v}_{ij} \times \nabla W_{ij}. \quad (2.26)$$

This factor effectively reduces the shear viscosity, but at the expense of not being able to vary the effective shear. In order to use the parameter α to control the effective shear viscosity, I do not apply the factor f_{ij} to the α term in the numerator of Π_{ij} . In the following subsection I present models of shocks using this hybrid artificial viscous term.

The set of hydrodynamical equations is closed with the inclusion of an equation of state. Most simply a polytropic form can be used: $P_i = K \Sigma_i^\gamma$, Σ being the surface density. The index γ is equivalent to the ratio of specific heats, and the constant K is the square of the isothermal ($\gamma = 1$) sound speed. More generally, the ideal gas law can be used in the form

$$P_i = (\gamma - 1) \rho_i u_i, \quad (2.27)$$

where u_i is the specific internal energy of particle i . The equation that governs the evolution of the specific internal energy is the first law of thermodynamics, $du = -P d\rho / \rho^2 + T ds$, where all nonadiabatic effects are included in the change in specific entropy, ds . The form of the thermal energy equation that is used in T2DSPH is the

same as in Hernquist and Katz (1989):

$$\frac{du_i}{dt} = \sum_j m_j \left(\frac{\sqrt{P_i P_j}}{\rho_i \rho_j} \right) \mathbf{v}_{ij} \cdot \nabla W_{ij} + \sum_j m_j \Pi_{ij} \mathbf{v}_{ij} \cdot \nabla W_{ij} + H + C. \quad (2.28)$$

The first term of this equation is the adiabatic term, the second term is the non-adiabatic viscous heating, and the final terms represent other non-adiabatic heating, H , and cooling processes, C , not associated with viscosity. If the polytropic equation of state is used, then the above equation is not needed. However, in this case, this equation is still integrated for each particle with only the adiabatic term included. This is done so that the amount of energy *lost*, due to the inclusion of an artificial viscosity with a adiabatic equation of state, can be measured by measuring the change in the total thermal energy. (See further discussion of this problem in section 2 of this chapter under adiabatic shock tests.)

Integration Method

Each particle has an intrinsic time step δt_i determined by its velocity, acceleration and a Courant-like condition:

$$\delta t_i = C \min \left(\frac{h_i}{v_i}, \left(\frac{h_i}{a_i} \right)^{1/2}, \frac{h_i}{h_i |\nabla \cdot \mathbf{v}|_i + c_i + 1.2(\alpha c_i + \beta \max_j |\mu_{ij}|)} \right), \quad (2.29)$$

where the parameter C is 0.3. The equations (2.17) are integrated using a time centered leap frog integrator, in which the velocities and positions are alternately stepped forward in time a half step out of synchronization. For an individual particle this can be written as

$$\begin{aligned} \mathbf{r}_i^{n+1/2} &= \mathbf{r}_i^n + \mathbf{v}_i^n \frac{\Delta t_i^n}{2} \\ \mathbf{v}_i^{n+1} &= \mathbf{v}_i^n + \mathbf{a}_i^n \Delta t_i^n \\ \mathbf{r}_i^{n+1} &= \mathbf{r}_i^{n+1/2} + \mathbf{v}_i^{n+1} \frac{\Delta t_i^n}{2}, \end{aligned} \quad (2.30)$$

the superscripts being the time step index. If the equations for all the particles are integrated simultaneously then the particle with the smallest time step will determine the time step for the entire system. In accretion disks with large central masses, the range in time steps can be large. To increase efficiency, multiple time steps are used, and particles are assigned an individual time step that is a power of two subdivision of the largest time step of the system:

$$\Delta t_i = \frac{\Delta t_{\text{sys}}}{2^{n_i}}; \quad \Delta t_{\text{sys}} = \max(\delta t_i). \quad (2.31)$$

Each particle, then, is assigned a time bin n_i so that $\Delta t_i \leq \delta t_i$. Particles are always allowed to move to a larger time bin (smaller time step), but may only move to a smaller time bin if that time bin is time synchronized with the particle's current time bin. Forces are then calculated for each particle i only once per Δt_i in order to step their velocities, while positions for all particles are stepped every $\Delta t_{\text{pos}} = \Delta t_{\text{sys}}/2^{n_{\text{max}}+1}$. Advancing the positions at every half step is necessary as forces need to be calculated at every half step. Care is also taken that an estimated velocity $\tilde{\mathbf{v}}_i$ is determined for all particles every Δt_{pos} , based on \mathbf{v}_i and the most recent estimate of \mathbf{a}_i , as the viscous forces depend on the local velocity field. For further details of implementing multiple time steps, see Herquist and Katz (1989).

Gravitational Softening

Gravitational softening is a common, and necessary, convention employed in N-body codes. Real gravitating systems often possess orders of magnitude more gravitating particles than can be modeled directly—the computational requirements are simply

too great. However, such a system can be approximated by using a smaller number of particles which interact via a softened gravity, which damps the effect of nearby encounters. In modeling stellar systems, such as galaxies, the ratio of the number of stars in a system being modeled over the number of gravitating particles employed in an N-body code, is often as many orders of magnitudes as the number of particles. When using an N-body code to model a gaseous system this difference between model and reality becomes essentially infinite, as a finite number of particles is now representing a continuous fluid.

The conventional method of softening is to use a form for the gravitational force between two particles which possesses a small constant term in the denominator:

$$\mathbf{F}_{ij} = \frac{m_i m_j \mathbf{r}_{ij}}{(r_{ij}^2 + \varepsilon^2)^{3/2}}, \quad (2.32)$$

where $G = 1$ is assumed throughout this work, and ε is the softening parameter. However, an alternative form of gravitational softening can be derived within the SPH formalism. The gravitational potential of a continuous mass distribution can be written as

$$\varphi(\mathbf{r}) = - \int \frac{\rho(\mathbf{r}') d\mathbf{r}'}{|\mathbf{r} - \mathbf{r}'|}. \quad (2.33)$$

Using the SPH estimate of the density [equation (2.9)] we have

$$\varphi(\mathbf{r}) = - \sum_j m_j \int \frac{W(\mathbf{r}' - \mathbf{r}_j, \varepsilon) d\mathbf{r}'}{|\mathbf{r} - \mathbf{r}'|} = \sum_j m_j I_j. \quad (2.34)$$

The integral I_j can be evaluated by noting that

$$\nabla^2 I_j = -4\pi W(\mathbf{r} - \mathbf{r}_j), \quad (2.35)$$

from Poisson's equation. Since the operator $(\nabla \cdot) = \frac{1}{r^2} \frac{d}{dr} r^2$ in spherical coordinates we have

$$r^2 \nabla I_j = - \int 4\pi W(\mathbf{r} - \mathbf{r}_j, \varepsilon) r^2 dr, \quad (2.36)$$

or using ∇_{u_j} instead of ∇_r , and $\nabla_r = \nabla_{u_j} \nabla_r u_j$, we have

$$\frac{u_j^2}{\nabla u_j} \nabla I_j = - \int_0^{u_j} 4\pi \varepsilon^3 W(u) u^2 du, \quad (2.37)$$

where ∇ represents ∇_r . From equation (2.29) we can write

$$\nabla \varphi = - \sum_j m_j \nabla I_j, \quad (2.38)$$

and using equation (2.31) we can write

$$\nabla \varphi = \sum_j m_j \frac{4\pi \varepsilon^3}{u_j^2} \left\{ \int_0^{u_j} W(u) u^2 du \right\} \nabla u_j, \quad (2.39)$$

where $\nabla u_j = \mathbf{u}_j / u_j$. Using the kernel

$$W(u) = \frac{1}{\pi \varepsilon^3} \begin{cases} 3/2(2/3 - u^2 + u^3/2), & \text{if } 0 \leq u \leq 1 \\ 1/4(2 - u)^3, & \text{if } 1 \leq u \leq 2 \\ 0, & \text{otherwise} \end{cases} \quad (2.40)$$

(equivalent to the kernel [equation (2.10)] with $h \rightarrow 2\varepsilon$), and evaluating the integral in the previous equation, a polynomial expression for the gravitational acceleration on particle i due to particle j can be written as $\mathbf{a}_{ij} = -m_j \mathbf{r}_{ij} g(r_{ij})$, where

$$g(r) = \begin{cases} \frac{1}{\varepsilon^3} \left[\frac{4}{3} - \frac{6}{5}u^2 + \frac{1}{2}u^3 \right] & 0 \leq u < 1 \\ \frac{1}{r^3} \left[-\frac{1}{15} + \frac{8}{3}u^3 - 3u^4 + \frac{6}{5}u^5 - \frac{1}{6}u^6 \right] & 1 \leq u < 2 \\ 1/r^3 & u \geq 2. \end{cases} \quad (2.41)$$

For the sake of completeness the contribution to the potential of a particle pair can be written $\varphi = -m_j f(r_{ij})$, where

$$f(r) = \begin{cases} -\frac{2}{\varepsilon} \left[\frac{1}{3}u^2 - \frac{3}{20}u^4 + \frac{1}{20}u^5 \right] + \frac{7}{5}\varepsilon & 0 \leq u < 1 \\ -\frac{1}{15}r - \frac{1}{\varepsilon} \left[\frac{4}{3}u^2 - u^3 + \frac{3}{10}u^4 - \frac{1}{30}u^5 \right] + \frac{8}{5}\varepsilon & 1 \leq u < 2 \\ 1/r & u \geq 2. \end{cases} \quad (2.42)$$

The above expressions are given by Herquist and Katz (1989) without derivation, citing Gingold and Monaghan (1978). This form of the softened gravity has the attractive feature that it is equivalent to Newtonian gravity when a particle pair's separation is greater than 2ϵ .

Accretion

During the hydrodynamic simulation an inner region, within a radius R_a of the central particle, is treated semi-analytically in order to circumvent the modeling of the central object (protostar), to avoid small time steps, and to define a radius at which particles are accreted onto the central object. The radius $R_a \simeq 0.05R_D$, and particles initially within this radius are kept and redistributed within R_a at each time step, so as to insure a continuous boundary at R_a , and thereby provide pressure support for the gas outside R_a . If, instead, no density profile within R_a is prescribed, but particles are simply removed from the simulation upon entering this inner region, the rate of accretion will be dependent on the size of R_a . In real disks, on the other hand, the accretion rate is determined by the combined action of any global non-axisymmetric modes present, and by viscous and magnetic forces acting locally throughout the disk. Since the physical condition of the disk determines the accretion rate, I wish the same to be true in the simulations.

While the outer particles in the disk ($r > R_a$) are evolved with SPH, inner particles are given an axisymmetric distribution so as to insure that the density and velocity are continuous across the boundary at R_a . This is done by first extrapolating from the outer disk the density Σ and the radial velocity V_r at R_a , which is accomplished by doing a linear regression on $\ln(\Sigma_i)$, and V_{ri} of particles in a small region around the inner

boundary. An estimate of V at R_a is made by assuming that the potential at R_a is due to the central mass and an exponential disk, with a scale length $0.25R_a$, in hydrostatic equilibrium. Now it is left to specify $\Sigma(r)$, $V(r)$, and $V_r(r)$ for the rest of the inner region. One of two different density distributions were used: a massless polytropic disk in a Keplerian potential, and an exponential disk. The initialization of these disks is described in Chapter 3.

The particles in the inner disk are used as neighbors in the SPH estimations for the particles located outside R_a to provide pressure support. To calculate the gravitational forces, however, these particles are ignored; the inner disk and the central object are treated as a single particle, with mass $M_c = M_c(\text{initial}) + (\text{accreted mass})$. The central region remains centered on the central particle which moves only under the gravitational influence of the outer disk, which is treated as a nongaseous particle during the simulation. Particles outside R_a are allowed to accrete into this region, at which point they are removed from the simulation and their masses are added onto M_c .

Treating the inner region in this semianalytical way introduces an inconsistency, in that the gaseous mass within R_a (M_i), determined from the density profile of equation (2.39) or (3-40), is not the same as the mass initially within R_a . Not only that, but M_i will change as the boundary values of Σ and V change. However, the only purpose of this inner disk is to provide a continuous inner boundary for the outer disk, rather than being an attempt to physically model the disk within R_a .

Summary of Code Parameters

A list of the parameters which control the performance of T2DSPH is listed for convenience in Table 2-1. The first parameter, N , determines how well a continuous

Table 2-1: Code Parameters

Parameter	Description	Value
N	Number of particles.	5131
ε	Gravitational smoothing parameter	$0.0272R_D$
θ	Tolerance angle	0.7 radians
\mathcal{C}	Courant number	0.3
R_a	Radius of inner region	$0.0544R_D$
ζ	Smoothing length adjustment parameter	1.3
α, β	Artificial viscosity parameters	—, 1.5

fluid is approximated by the numerical method. While a large number of particles results in a better approximation, the cost is greater computational time per time step. The next two parameters control aspects of the gravitational force calculations, while the Courant number determines the size of the time integration steps. The fifth parameter defines the radius about the central massive particle at which SPH particles are removed from the simulation, and their masses added to the central particle. The parameter ζ determines the average number of neighbors, which should be greater than 13 for two dimensions, and greater than 35 for three dimensions. The last two parameters are the artificial viscosity parameters; β is not varied, while α is given one of three values to introduce varying degrees of effective shear viscosity. In this sense α is treated as a physical parameter of the gas. The nominal values used for the simulations presented in Chapter 4 and 5 are also

given. The units of length are expressed as fractions of the initial disk radius, R_D . Tests verifying that the chosen values result in accurate simulations are presented in Chapter 4.

Tests

To validate the accuracy of the hydrodynamics code, or at least to increase confidence in its results, a variety of test models were run which can be compared to analytical expectations. The first model presented is that of a stable, rotating Maclaurin disk. A series of two-dimensional adiabatic shock fronts were also modeled to specifically test the hybrid viscosity employed in T2DSPH (see previous section of this chapter).

Maclaurin Disk

A test to the code, using both SPH and self-gravity, is a simulation of a Maclaurin disk, the two-dimensional counterpart to a Maclaurin spheroid, which represents a polytropic stable solution to Poisson's equation and hydrostatic equilibrium when $\gamma=3$. The surface density profile is $\Sigma(r) = \Sigma_o \sqrt{1 - (r^2/R_D^2)}$, and is initialized by radially stretching a regular grid of particles (see Chapter 3).

Figure 2-2 shows the surface density and velocity profile after 26 dynamical times ($T_{dynamic} = R_D/V(R_D)$), the dotted line representing the analytical solution. The particle velocities lie slightly below the analytical solution because the code employs softened gravity. Because the disk is in solid body rotation the effective shear viscosity of the code has no effect on the disk during the simulation, that is, angular momentum transfer is not induced.

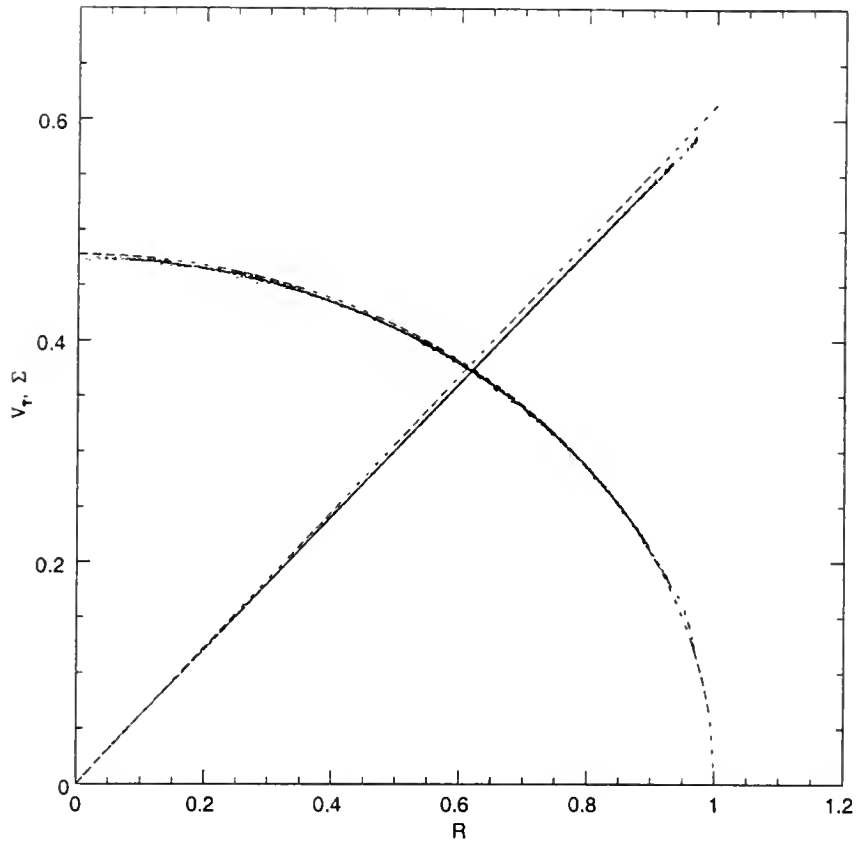


Figure 2-2: Surface density and velocity profile of the Maclaurin disk after 26 dynamical times.

Two-Dimensional Shocks

Two-dimensional adiabatic shocks, without self-gravity, were modeled as a means of specifically testing the implementation of SPH in the program T2DSPH, and as a test of the hybrid viscosity that is employed (see previous section). A two-dimensional sheet, with an initial density equal to one, was initialized by placing particles on a regular grid. Particles with a positive x coordinate are given an initial negative x component velocity of Mach 1, and those with a negative x coordinate are given a Mach 1 velocity in the opposite direction. This results in the formation of an accretion shock, having two boundaries parallel to the y axis traveling away from each other. The sheet has a

finite extent, resulting in rarefaction waves traveling inward from the edges of the sheet. However, for a time, the central region of the sheet will be uncontaminated by these waves, allowing the simulation there to be compared with an analytical solution.

For all the tests presented here the polytropic equation of state was used, with $\gamma = 2$. From the jump conditions of an adiabatic shock, and using the polytropic equation of state, a solution can be found for the shock velocity and post-shock density. However, this will not be the desired solution. By employing an artificial viscosity, at the same time that we use a polytropic equation of state, we have introduced an effective *cooling*, because the kinetic energy dissipated by viscosity is not added to the system as heat. This thermal energy would be added to the material if the specific thermal energies of the particles, u_i , were evolved:

$$\left(\frac{du_i}{dt} \right)_{\text{non-adiabatic}} = \frac{1}{2} \sum_j m_j \Pi_{ij} v_{ij} \quad (2.43)$$

In other words, in spite of our polytropic equation of state, we do not have a purely adiabatic shock. This can be taken into account in our analytical solution by the addition of a post-shock cooling term, Q (heat lost per unit mass), in the energy jump condition. Therefore, using the polytropic equation of state, with $K=1$, the jump conditions are:

$$\begin{aligned} \rho_1 v_1 &= \rho_2 v_2, \\ \rho_1 v_1^2 + \rho_1^\gamma &= \rho_2 v_2^2 + \rho_2^\gamma, \\ \frac{\gamma}{(\gamma-1)} \rho_1^{\gamma-1} + \frac{1}{2} v_1^2 &= \frac{\gamma}{(\gamma-1)} \rho_2^{\gamma-1} + \frac{1}{2} v_2^2 + Q, \end{aligned} \quad (2.44)$$

which results in the final solution:

$$v_2 = \frac{-b + \sqrt{b^2 + 4[(\gamma - 1)(v_o^2/2 - Q) + \gamma\rho_1^{\gamma-1}]}}{2}, \quad (2.45)$$

$$\frac{\rho_1}{\rho_2} = \frac{v_2}{v_o + v_2},$$

where

$$b = \frac{3 - \gamma}{2}v_o + \frac{\gamma - 1}{v_o}Q. \quad (2.46)$$

The solution above is given in the post-shock reference frame, with v_o and v_2 representing the pre-shock velocity and the shock front velocity respectively.

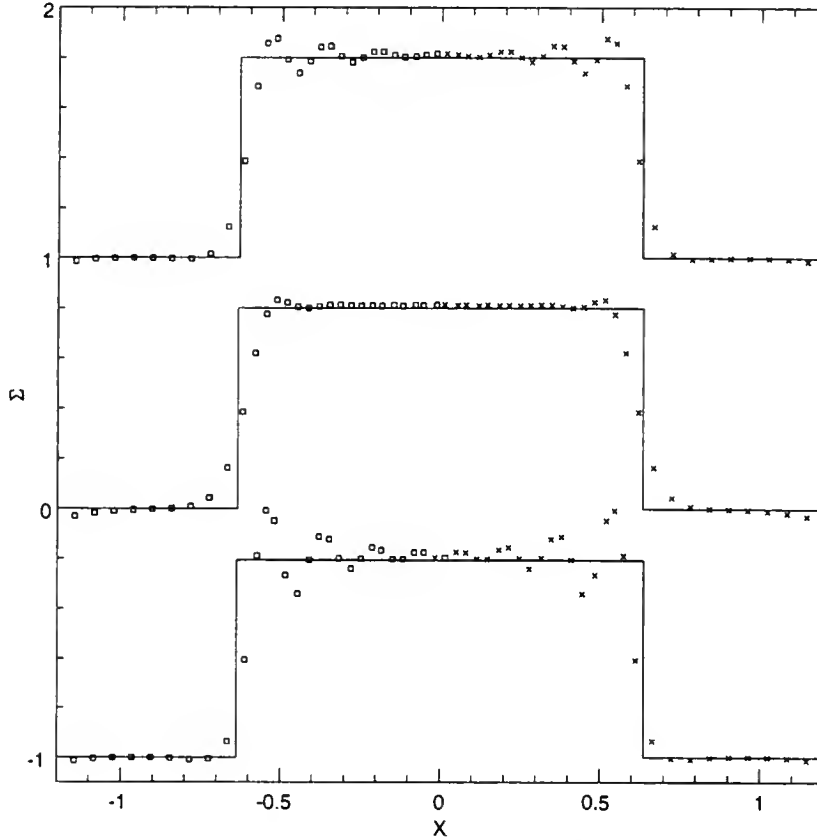


Figure 2-3: Surface density of three two-dimensional accretion shocks. The bottom accretion shock's surface density is offset from the top by -2 , and the middle by -1 , from the top accretion shock.

In Figure 2-3 three simulations are shown. The bottom two, with $\alpha = 0.25$, 1, in ascending order, and $\beta = 0$, do not use the viscosity switch. In the third simulation, shown at the top of the figure, the hybrid viscosity is employed, with $\alpha = 0.25$, $\beta = 1.5$. Although this solution is not as good as the first, it has much less effective shear viscosity. Also, notice that there is no interpenetration of particles, as there is in the first case.

Analysis of Disks

Evaluation of Effective Shear Viscosity

In the SPH formalism shear viscous forces are not introduced directly, but instead are a result of the artificial viscous term, a^{visc} , being evaluated over a finite region. Hernquist's form of the artificial viscous term illustrates this more explicitly; being dependant on $(\nabla \cdot \mathbf{v})$ it formally introduces no shear force but is purely a bulk viscous force. However, what is used numerically is an estimate of $(\nabla \cdot \mathbf{v})$, which is evaluated by looking at particles within a finite region (neighborhood) around the point of interest. In addition, the total force on a particle is a sum of symmetric forces between itself and neighboring particles at other radii, and the tangential components of these forces transfer angular momentum. These errors in the estimation formally vanish as the smoothing length approaches zero.

This implicit introduction of the shear viscous forces in SPH causes an unfortunate situation for those attempting to use this method to model astrophysical disks, where large velocity gradients can exist and shear viscous forces play an important role. In SPH both bulk and shear viscous forces are effectively coupled together, rather than

being independently parameterized. In an attempt to decouple the bulk viscosity needed to describe shocks and the effective shear introduced by the artificial viscous term, an alternate form for the artificial viscous term was introduced in section 2. Here I wish to address how the effective shear viscosity scales with the viscous parameter α or any other quantities. The functional form of the artificial viscous term suggests that, like the α_ν viscosity of accretion disk theory (Shakura & Sunyaev 1973), the effective shear coefficient scales as $\alpha_\nu ch$, c being the sound speed and h representing a characteristic length. Artymowitz (1993) has suggested that for SPH $\alpha_\nu \simeq 0.1\alpha$, with the characteristic length being considered as the smoothing length. While the similar functional form of the artificial viscous term and the viscosity of accretion disk theory is suggestive, it is no guarantee that the implicit shear will behave similarly, being a by-product of estimation. It is therefore desirable to determine not only the relative magnitudes of α and α_ν , but also whether the effective shear scales in the way that the artificial term suggests.

To investigate the effective shear associated with the artificial viscosity three models were run with $\alpha = 1, 0.5, 0.25$. An exponential disk was initialized, with the velocity field now consistent with assuming that the disk is in a Keplerian potential with no pressure support. These disks were evolved with SPH consistent with these assumptions, and they remain axisymmetric in the absence of self-gravity. Since no pressure support is required it was found to be advantageous to exclude the inner disk, and to simply remove particles from the simulation when they approached within R_a of the gravitating particle located at the origin. After a few dynamical times a radial velocity flow is established throughout most of the disk as a result of the effective shear viscosity.

From standard accretion disk theory the radial velocity profile of an axisymmetric thin disk, with no radial pressure support, is given by

$$v_r = \left[\Sigma r (\Omega r^2)' \right]^{-1} \frac{\partial}{\partial r} (\Sigma \nu r^3 \Omega'), \quad (2.47)$$

where Ω is the angular velocity, and the primes denoting radial derivatives (Pringle 1981).

If the shear coefficient ν is not a constant, then

$$\nu = (\Sigma r^3 \Omega')^{-1} \int_0^r v_r \Sigma r (\Omega r^2)' dr. \quad (2.48)$$

For a Keplerian potential $\Omega^2 = GM/r^3$, which upon substitution results in

$$\nu = -\frac{1}{3} (\Sigma r^{1/2})^{-1} \int_0^r v_r \Sigma r^{1/2} dr. \quad (2.49)$$

If in SPH the shear coefficient, ν , scales as $\alpha_\nu ch$, then $\alpha_\nu = \nu/ch$ will be constant over the disk, and can be estimated by evaluating the above expression for ν . The SPH particles of the evolved disks are used to find the surface density and radial velocity of annuli centered on the origin, and the integral in equation (2.48) is estimated by the sum over the annuli

$$\sum_i v_{r,i} \Sigma_i r_i^{1/2} \Delta r, \quad (2.50)$$

where Δr is the width of the annuli and is equal to $.05R_D$. Now at each radius r_i the shear viscous parameter α_ν can be estimated by using

$$\alpha_\nu(r_i) = \frac{\nu_i}{1.3 \sqrt{2K m_p}}, \quad (2.51)$$

remembering that the sound speed $c = \sqrt{2K\Sigma}$ when $\gamma = 2$, and the smoothing length $h = 1.3 \sqrt{m_p/\Sigma}$, where m_p is the mass of each SPH particle and assuming that the

smoothed density $\Sigma^* = \Sigma$. Using this method on the three models described above yielded α_ν approximately constant with radius, with values of 0.1, 0.05, 0.02 \pm .01 for each model respectively. Much of the uncertainty arises from a time variability that is probably due to radial oscillations. The fact that α_ν is found to be approximately constant over the disk confirms that the effective shear viscosity scales the same way as the artificial (bulk) viscosity.

Evaluation of Modes and Their Frequencies

To follow the evolution of the nonaxisymmetric modes a method similar to that of Papaloizou and Savonije (1991, hereafter PS) is adapted; a polar grid, concentric with the center of mass of the system, is imposed on the disk with a radial extent from 0.1 to 1. The density is evaluated at 64 points, equally spaced in azimuth, for each of 25 equally spaced radii, and then normalized by dividing the density at each point by the average density at each radius. The modes are identified by doing a Fourier analysis of the normalized density in azimuth at each radius, and plotting the power spectrum as a function of mode number and radius (Figure 2-4). Unlike PS, the density is not taken to be the mass/area of individual cells defined by the grid, but instead the SPH formalism is used to estimate the density at each grid point:

$$\Sigma_k = \sum_i m_i W_{ki}, \quad (2.52)$$

where the sum is over all particles having the grid point k within their smoothing length h_i . This analysis can be done at each time that the position and smoothing lengths of the particles are output by the SPH code, which is at every 0.5 dynamical times (see Chapter

3 on choice of units). Another useful way of presenting the results of this analysis is with a contour plot of the dynamic spectrum: the power in a particular mode as a function of radius and time (Figure 2-5). The evolution of a particular disk can be more simply characterized by considering the maximum power, with respect to radius, of each mode as a function of time (Figure 2-6).

To find the frequencies that are present, the SPH code outputs the estimated density at 100 points along both the x and y axes of a Cartesian grid, with its origin at the center of mass of the system, at every third system time step. After the simulation a Fourier analysis is done in time, at each radius, for a specified time interval. Contour plots of the power, as a function of radius and frequency, can be displayed separately for each axis, or averaged together (Figure 2-7). Provided that there are a limited number of modes, the contour plots of the modes and of the frequencies can be used together to match frequencies with particular modes.

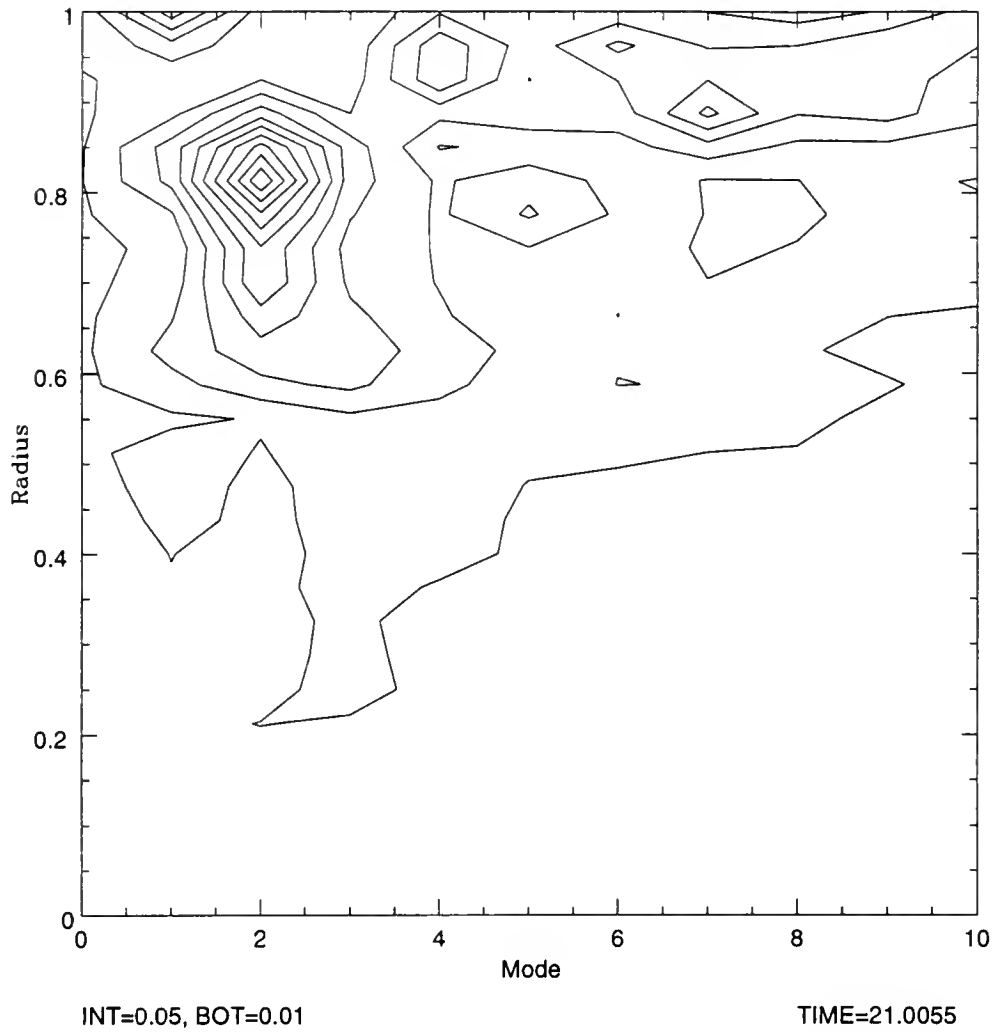
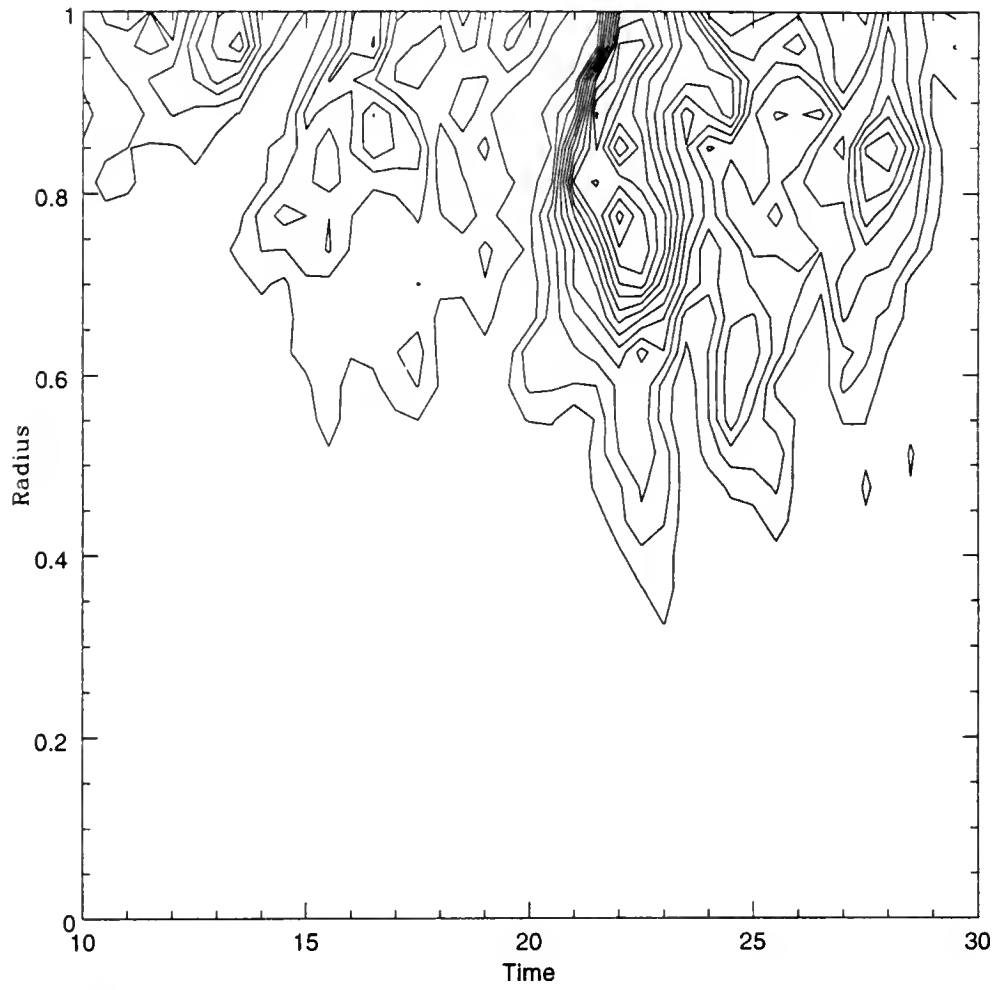


Figure 2-4: Normalized power spectrum of modes of model A2 (see Chapter 4) at Time = 21 dynamical times.



INT=0.05, BOT=0.05

Figure 2-5: Dynamic power spectrum of the normalized density for mode = 2 of model A2.

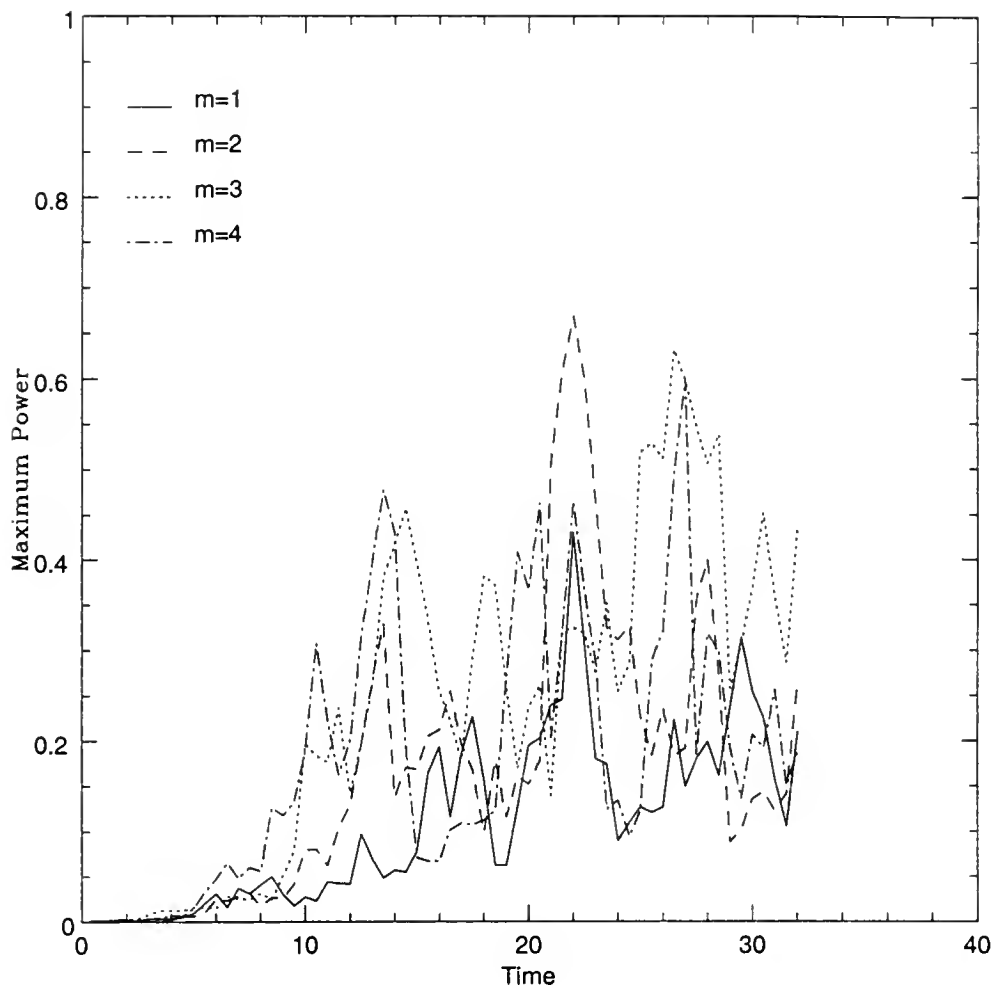
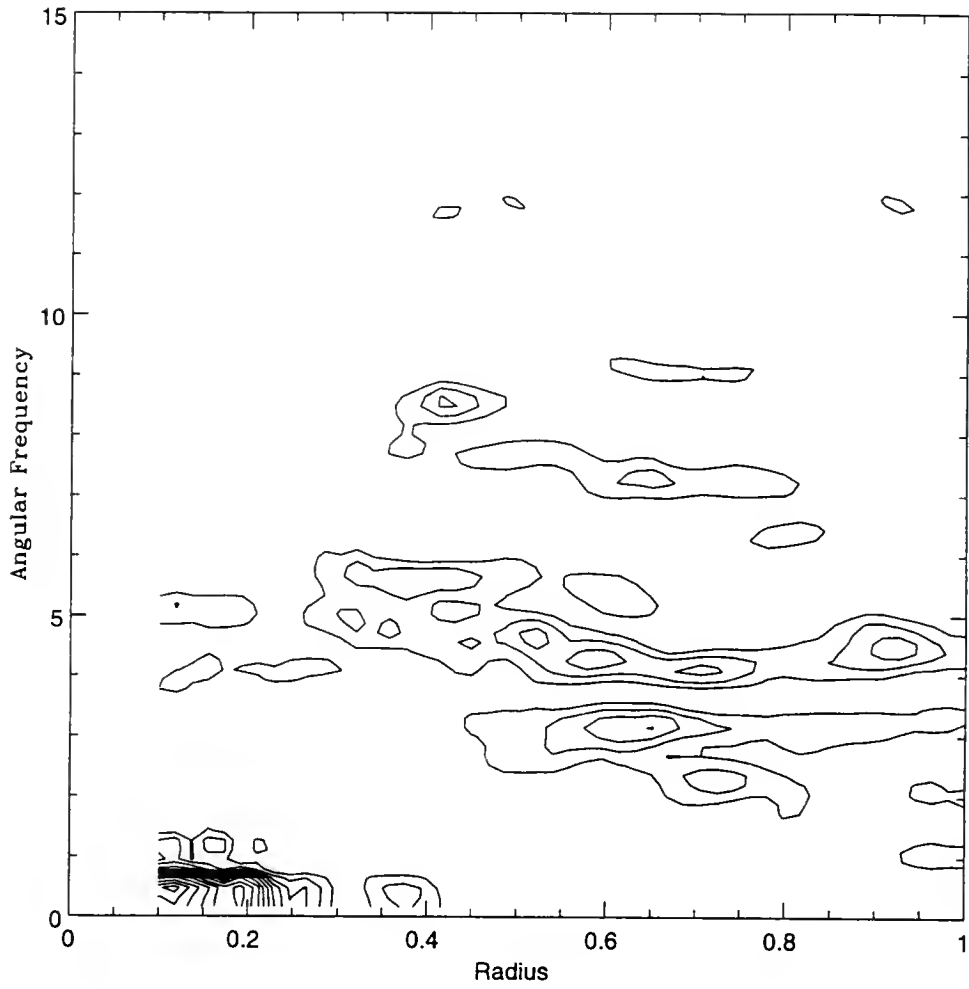


Figure 2-6: Maximum of the power spectrum, in modes = 1 - 4, of the normalized density in model A2.



BOT=13.2012=99.9% conf.

Time = 20.0 to 30.0

Figure 2-7: The average Lomb periodogram of normalized density temporal fluctuations along Cartesian axes for model A2.

CHAPTER 3 INITIALIZATION OF DISKS

In this chapter the initialization of the disks numerically evolved with SPH is detailed. Initializing a disk for an SPH program involves finding a distribution of points in phase space which suitably describes the particular density and velocity field of the disk. Except for perturbations, all initial disks used in this work are axisymmetric and in radial hydrostatic equilibrium. Global parameters describing a disk are the disk mass M_D and the radius of the disk R_D . With the exception of the Maclaurin disk, and in the spirit of modeling protostellar accretion disks, there will also be a central particle with mass M_c , giving a total mass for the protostar/disk system of $M_T = M_c + M_D$. Treating the central object as a point mass is equivalent to assuming it is several magnitudes smaller in size than the disk radius. This assumption is valid for protostellar disks that have radii approximately $10 - 1000 AU$. Parameters of the gas itself are the ratio of specific heats, γ , and the isothermal sound speed K . Unless stated otherwise, dimensionless units of mass, length, and time will be used for which $M_T = R_D = G = 1$. In this case the unit of time then becomes the dynamical time: $T_D = R_D/V(R_D) = (R_D^3/GM_T)^{1/2}$. The models may then be scaled to the desired dimensions. For instance, if $M_T = 1M_\odot$ and $R_D = 100AU$ then the dynamical time is 159 years.

Two-dimensional Disks

Maclaurin Disk

A two-dimensional disk has already been presented in Chapter 2 as one of the tests to the SPH code: the Maclaurin disk. This is a two-dimensional version of the Maclaurin spheroid, which represents a polytropic stable solution to Poisson's equation and hydrostatic equilibrium when $\gamma=3$. The surface density profile of the Maclaurin disk is

$$\Sigma(r) = \Sigma_o \sqrt{1 - (r^2/R_D^2)}, \quad (3.1)$$

and is initialized by radially stretching a regular grid of particles. The central density Σ_o is given by $3M_D/2\pi R_D^2$.

This method of constructing an axisymmetric disk I call the stretched grid method, and is equivalent to transforming the radial coordinate, R , of a particle in a uniform disk to the new radial coordinate r via a coordinate transform function: $r = A(R)R$. To apply this method the transform function, or stretching factor, $A(R)$, must be found which will transform a uniform disk into a Maclaurin disk. The desired function can be found by considering an invariant of the transform, the mass within a given radius. That is, the mass within a given radius of the uniform disk, $M_D R^2/R_D^2$, is equivalent to the mass within $r = AR$ of the Maclaurin disk, which is

$$M_D \left[1 - \left(1 - \frac{r^2}{R_D^2} \right)^{3/2} \right]. \quad (3.2)$$

Setting the above expression equal to $M_D R^2 / R_D^2$, replacing r with AR , and solving for A gives the desired transform function:

$$A(R) = \frac{R_D}{R} \left[1 - \left(1 - \frac{R^2}{R_D^2} \right)^{2/3} \right]^{1/2}. \quad (3.3)$$

Giving each particle in the uniform disk with radial coordinate $R_i < R_D$ a new radial coordinate $r_i = A(R_i)R_i$, and discarding any remaining particles, transforms a uniform grid into a Maclaurin disk (see Fig 3-1).

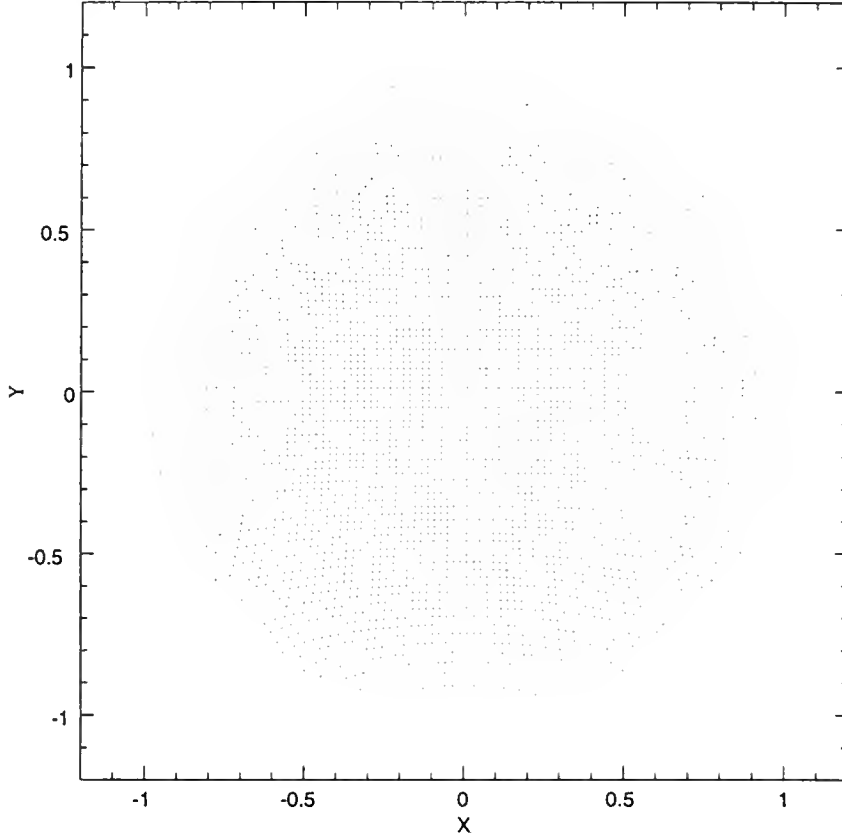


Figure 3-1: Initial distribution of particles for the Maclaurin Disk.

This method, though quite general, was found to be of limited usefulness when it was used to construct disks that possess stronger central mass concentrations, such as the

polytropic Keplerian and exponential disks described below. The undesirable feature of the disks built with this method is the presence of artifacts of the grid from which the disk was stretched: the direction to the nearest neighbor is not isotropic on the average, the particles instead appearing to be “planted” in rows, and the edge of the disk is corrugated, or scalloped. These artifacts are not severe in the Maclaurin disk shown in Fig 3-1, but they become much more pronounced in centrally condensed disks.

Once the particles have been spatially distributed their velocities are determined. The initial tangential velocity of each particle V_i in the Maclaurin disk is $kV_o(r_i)$, $V_o(r_i) = Cr_i/R_D$ being the initial tangential velocity required if the disk were supported purely by rotation, with $C^2 = 3\pi GM_D/4R_D$. The rotation parameter $k = V/V_o = 0.4$ is assumed to be a constant with respect to radius, and gauges the importance of rotational support. Maclaurin disks with $k < 0.5$ will be stable against secular instabilities (Binney and Tremaine, 1987). The tangential velocity V_o was found from the gravitational force on each particle, evaluated using a direct sum over all other particles, rather than using the analytical formula for V_o , because of the softened form of gravity employed. From the requirement of radial hydrostatic equilibrium,

$$\frac{1}{\Sigma} \frac{\partial P}{\partial r} = \frac{V^2}{r} - \frac{V_o^2}{r}, \quad (3.4)$$

which follows from the radial component of the equation of motion in cylindrical coordinates, and the assumptions of axisymmetry and no radial motion, the constant K in the equation of state is found. It is determined to be a function of the disk parameters:

$$K = (1 - k^2) \frac{G\pi^3 R_D^3}{9M_D}. \quad (3.5)$$

The above expression is found from assuming unsoftened Newtonian gravity, whereas the code that will evolve the initial conditions uses softened gravity. Because the value of the softening parameter is small, the error introduced by this inconsistency is also small. Using $k = 0.4$ and $M_D = R_D = 1$ gives us $K=2.8939$.

Exponential Disks

Exponential disks were chosen to simulate accreting protostellar disks in large part due to the previous theoretical and numerical work that has been done with such disks (Papaloizou and Savoneji, 1991). This is the practical reason. That such a choice is reasonable physically is somewhat born out by the numerical result that the exponential profile is preserved over most of the disk for the majority of the duration of the numerical evolution, at least in the case where the central star to disk mass ratio was equal to three. The models with equivalent star and disk masses differ significantly in that a $m=1$ mode completely dominates the mass distribution of the disk and erases all axisymmetry. In any event, angular momentum transport and mass accretion reinforce the centrally condensed mass distribution.

To represent the mass distribution of these disks, particles representing the gas, all of equal mass, are placed in concentric rings around the central particle at the origin. The surface density profile,

$$\Sigma(r) = \Sigma_0 \exp(-r/r_s), \quad (3.6)$$

is achieved by appropriately spacing concentric rings, from the inside out, so that the correct interparticle separation, $(m/\Sigma(r))^{1/2}$, is nearly imposed. The scale length r_s is

set to 0.25 in all models, while the ratio of the central to disk mass M_c/M_D is either 3 or 1. The central density is then given by the expression

$$\Sigma_o = \frac{M_D}{2\pi r_s^2} \left[1 - e^{-R_D/r_s} \left(1 + \frac{R_D}{r_s} \right) \right]^{-1}. \quad (3.7)$$

Before the disk is built the number of total desired particles, N_{tot} , is specified. The gaseous particles are then assigned a particle mass of $m_p = M_D/N_{\text{tot}}$. The first ring of gaseous particles consists of six particles placed at equal angular intervals around the central particle at a distance $0.5 \times r(6m_p)$, one half the radius within which a mass of $6m_p$ resides in. Here, and in what follows, the radius $r(m_r)$ is found by using the Newton-Raphson method, with the expression for the mass that lies interior to a given radius for the exponential disk:

$$m(r) = \Sigma_o 2\pi r_s^2 \left[1 - e^{-r/r_s} (r/r_s + 1) \right]. \quad (3.8)$$

For subsequent rings, the ring radius r_n is determined by finding the number of particles in the ring, N_n , that minimizes the quantity

$$\left| (r_n - r_{n-1}) - \frac{2\pi r_n}{N_n} \right|, \quad (3.9)$$

where $r_n^2 = (r_n'^2 + r_{n-1}'^2)/2$, with each primed radii being the outer radius of the annulus about the n th ring. This is an attempt to equate the distance between ring n and $n+1$ with the distance between particles in ring n . To calculate the quantity above, the radius of an n th ring with N_n particles is found from r_{n-1}' , and the radius r_n' within which there are $N_r = \sum_{j=1}^n N_j$ particles. This iterative procedure is continued until $N_r \geq N_{\text{tot}}$. If $N_r > N_{\text{tot}}$ then N_{tot} is set equal to N_r , the particle mass m_p is reinitialized, and the disk

rebuilt. For an exponential disk this procedure will converge on a value of N_{tot} which will result in a disk being built with $N_r = N_{\text{tot}}$ at the outer most ring. In the case where 5000 particles are initially requested, the above algorithm results in an exponential disk with 5130 particles. Though the above procedure for building an axisymmetric disk with particles on concentric rings is in principle quite general, requiring only that the desired function $m(r) = 2\pi \int_0^r \Sigma(u)u du$ be specified, it is not known whether the iteration will converge for other density profiles.

An initial circular velocity V is assigned for each ring of particles according to the equation of radial hydrostatic equilibrium, which leads to

$$V^2 = \gamma K r \Sigma^{\gamma-2} \frac{d\Sigma}{dr} + V_o^2 = -\gamma K \frac{r}{r_s} \Sigma^{\gamma-1} + V_o^2, \quad (3.10)$$

where V_o^2/r for each ring is calculated by finding the radial gravitational acceleration for each particle, through a direct summation, and averaging the radial components of the acceleration of the particles in each ring. In doing this calculation softened gravity is used. The value of the constant K , the square of the isothermal sound speed, is determined by assuming radial hydrostatic equilibrium throughout the disk and specifying a minimum value for Toomre's stability parameter $Q = \kappa c / \pi G \Sigma$, where $\kappa = (\frac{1}{r} \frac{dV_o^2}{dr} + 2 \frac{V_o^2}{r^2})^{1/2}$ is the epicyclic frequency. Toomre's parameter is a local stability parameter, derived from the linearized hydrodynamic equations of a local region of a rotating gas sheet, which describes the stability of a two-dimensional disk (Toomre 1964). When $Q > 1$ the disk is stable against axisymmetric perturbations, but the disk generally remains unstable to non-axisymmetric modes for values of $1 < Q < 3$. The epicyclic frequency, κ , is numerically

evaluated for each ring n with the expression

$$\kappa_n^2 = \frac{1}{r_k} \frac{(V_o^2(r_{n+1}) - V_o^2(r_{n-1})))}{(r_{n+1} - r_{n-1})} + 2 \frac{V_o^2(r_n)}{r_n^2}. \quad (3.11)$$

Using the expression $c^2 = \gamma K \Sigma^{\gamma-1}$, Toomre's stability parameter can be written as

$$Q = \frac{\kappa (\gamma K \Sigma^{\gamma-3})^{1/2}}{\pi G}. \quad (3.12)$$

An expression for the constant K can be found from specifying Q_{\min} using the previous equation:

$$K = \frac{(Q_{\min} \pi G)^2}{\gamma \min_k \left(\kappa_k^2 / \Sigma_k^{3-\gamma} \right)}. \quad (3.13)$$

For all the two-dimensional exponential disks, Q_{\min} is set to 1.15. In Figure 3-2 is shown the initial velocity profile and the value of the Toomre Q parameter with respect to radius.

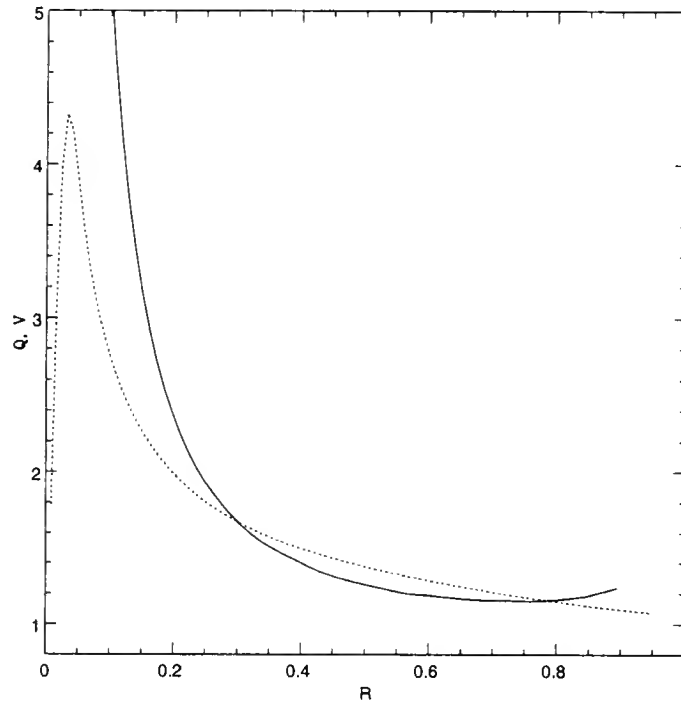


Figure 3-2: The initial Toomre Q parameter (solid line) and tangential velocity (dotted line) with respect to radius for a $M_c/M_D = 3$ disk.

A density perturbation is then seeded into the disk by adding independent Gaussian noise to the x and y coordinates of each particle, the displacement having a standard deviation of 0.02 of the interparticle distance. This perturbation is added after the tangential velocity V is determined for all of the rings. The subroutine which generates the Gaussian noise is taken from *Numerical Recipes* (Press *et al.*, 1992). This introduces a density fluctuation with an rms amplitude 0.03 times the average density. The purpose of introducing the noise is twofold. The first is to excite all possible modes, so as to observe which modes grow fastest and those which tend to dominate the disk. This was the primary reason for the introduction of noise. The second, more serendipitous, is to avoid a flaw in the initializing method. If the SPH particles were distributed on concentric rings, without noise, then the regularity of their distribution will suppress some possible responses while enhancing others. For instance, it was found from evolving a disk with no noise, but with an $m=2$ perturbation (see below), that the initial growing mode developing early in the disk's evolution, while particles still lay on concentric rings, led to the rings being distorted, and themselves becoming structures that excited a response. The folded and compressed portions of the rings formed shock "wakes" to the arms.

The FORTRAN code which generates the initial positions and velocities of the particles that describe the exponential disk is listed in Appendix C. The initial distribution of points for the exponential disk is shown in Figure 3-3, while the initial radial profile of the density is shown in Figure 3-4.

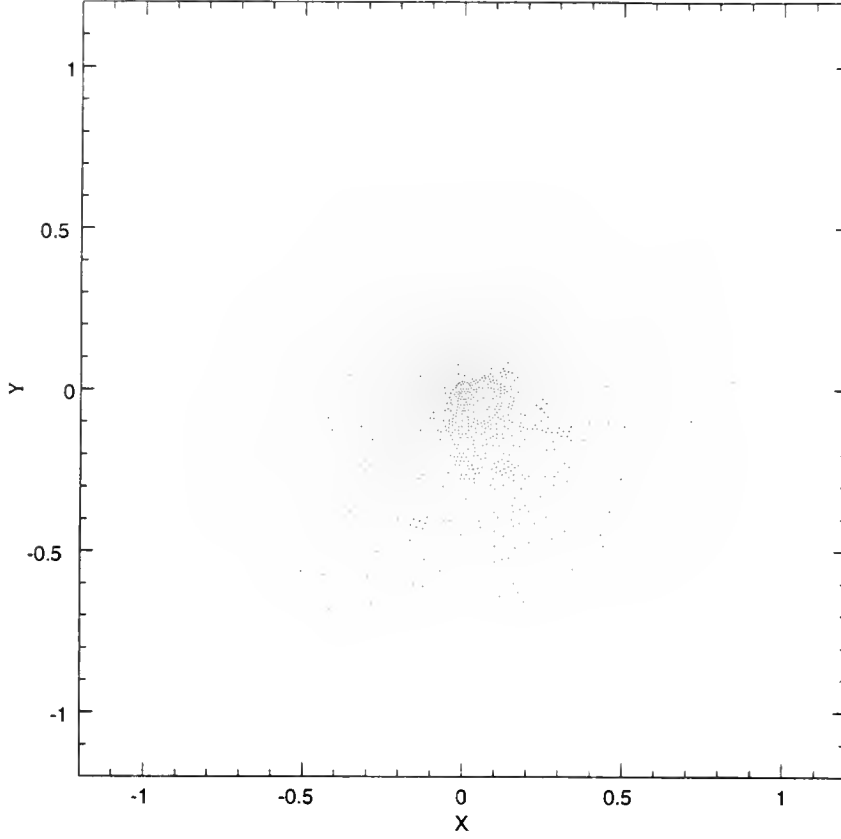


Figure 3-3: Initial distribution of particles in an exponential disk.

In one model an $m=2$ density perturbation is introduced of the form $\Sigma'(r, \varphi) = \Sigma_o(r)p(r)\cos(m\varphi)$, where $p(r) = A_p[\pi(r - r_s)/(R_D - r_s)]$, with A_p being 0.01, and Σ_o representing the unperturbed density given by equation (3.5). This perturbation is induced by displacing the particles along the circumference of their respective ring. To find the displacements which will effect the desired density perturbation in a particular ring, I consider the perturbed density in the form $\Sigma'(k, x) = A\cos(kx)$, where $A = \Sigma_o(r)p(r)$, $k = m/r$, and $x = \varphi r$, an arclength. This form of the perturbed density can be considered

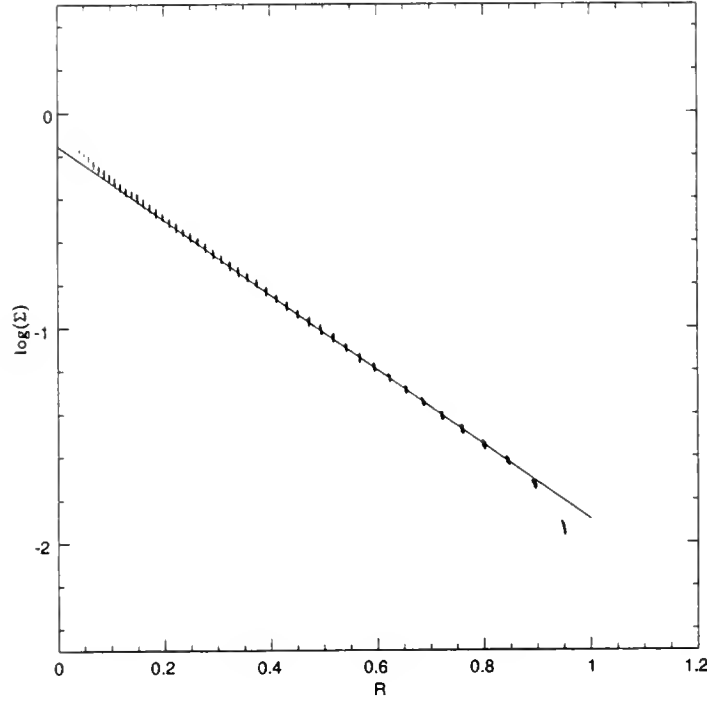


Figure 3-4: Initial density profile of an exponential disk with $M_D = 0.25$ and a scale length of $r_s = 0.25$. Each point corresponds to the estimated density at each particle position, evaluated with the SPH method, while the solid line is the intended density profile.

as being the perturbed density of a one-dimensional sound wave at time equal zero, which is more generally written as $\Sigma'(x, t) = A \cos(kx - \omega t)$. Linearization of the one-dimensional equation of continuity and motion gives the wave equation

$$\frac{\partial v'}{\partial t} = -\frac{c_o^2}{\Sigma_o} \frac{\partial \Sigma'}{\partial x}, \quad (3.14)$$

where c_o is the sound speed equal to ω/k . Using this equation and the given form of the perturbed density, I find the perturbed velocity $v' = c_o p(r) \cos(kx - \omega t)$. From this perturbed velocity a displacement can be found:

$$\Delta x = \int v' dt = -\frac{p(r)}{k} \sin(kx - \omega t). \quad (3.15)$$

The desired displacements are those at $t = 0$, at which time an equation for the perturbed particle position x is formed:

$$x = x_o - \frac{p(r)}{k} \sin(kx), \quad (3.16)$$

x_o being the unperturbed position, and $\Delta x = x - x_o$. This equation is solved for each particle using the Newton-Raphson method.

To find the associated perturbed velocities, the velocities of the one-dimensional sound wave employed above are not used. While the above derivation is sufficient for finding the displacements, it is an oversimplification with regards to the gas dynamics of the disk. Instead, the equation of radial hydrostatic equilibrium is used to find the tangential velocity $V(r, \varphi)$ by inserting the perturbed density. This results in the expression

$$V^2(r, \varphi) = V_o^2 - \gamma K \Sigma_o^\gamma \frac{r}{r_s} \left\{ 1 - p(r) \cos(m\varphi) \left(1 + \left(\frac{\pi}{1 - r_s} \right) \cot \left[\frac{\pi(r - r_s)}{1 - r_s} \right] \right) \right\}. \quad (3.17)$$

Another model that was evolved includes an encountering particle, with a mass of 0.5, that approaches the central particle with a minimum distance of roughly 1.0. Otherwise the disk and central particle are initially identical to model D2. The initial separation between the encountering particle and central particle is about 8.772. The energy of the encountering particle is specified to be

$$E = \frac{1}{2} \frac{GMm_e}{a}, \quad (3.18)$$

where M is the mass of the central object and disk (1.0), m_e is the mass of the encountering particle (0.5), and a is the separation. With this energy the relative velocity

of the particle is $v = \sqrt{3GM/a}$. A common parameter to describe an encounter is the impact parameter b . Instead of specifying this parameter I have specified the minimum separation $a_{\min} = 1.0$. The impact parameter can then be determined as

$$b = \frac{v_{\max} a_{\min}}{\sqrt{\frac{2}{m_e} E}}, \quad (3.19)$$

with

$$v_{\max} = \sqrt{\left(E + \frac{GMm_e}{a_{\min}}\right) \frac{2}{m_e}}. \quad (3.20)$$

The energy of the encountering particle can also be parameterized in terms of the velocity of the particle if removed to infinity, $V_o = \sqrt{\frac{2}{m_e} E}$. The specific angular momentum of the encountering particle is then bV_o . Using this, the components of the velocity can then be determined: $v_T = bV_o/a$, and $v_a^2 = v^2 - v_T^2$, being the velocity tangential and parallel to the separation vector respectively.

Inner Disks

As described in Chapter 2, a small inner region about the central particle is described analytically, rather than being evolved with SPH. Two different inner disks are used. If the disk mass within R_a is much less than M_c , it may be considered as a massless Keplerian disk. Assuming axisymmetry in two dimensions, radial hydrostatic equilibrium, and using a polytropic equation of state with $\gamma = 2$, the density distribution within R_a may be written

$$\Sigma(r) = \frac{1}{2} \frac{GM_c}{K} (1 - k^2) \left(\frac{1}{r} - \frac{1}{R_a} \right) + \Sigma(R_a). \quad (3.21)$$

It has also been assumed that $k = V/V_o = V(R_a)/V_o(R_a)$ is a constant throughout the inner disk, and V_o is the circular rotation without pressure support. The inner particles are

repositioned on concentric rings to describe this density distribution, with their masses being reassigned so that m_p is equal to M_{in} divided by the number of inner particles, where M_{in} is the mass interior to R_a as determined from $\Sigma(r)$ in the equation above. Using $V_o = \sqrt{GM_c/r}$ for the inner disk, the tangential velocities $V_i = kV_o(r_i)$ are determined. The radial velocity is specified by assuming that the mass accretion rate $\dot{m} = 2\pi\Sigma(r)V_r(r)$ is constant throughout the inner disk. Since the values of Σ , V , and V_r at the boundary are determined at each time step, the inner disk must be reinitialized at each time step as well.

The inner disk described above is specifically for $\gamma = 2$; for other values of γ an alternate model of the inner disk must be used. The second inner disk that is used is one with an exponential density profile,

$$\Sigma(r) = \Sigma_o e^{-r/r_s}. \quad (3.22)$$

The scale length is found from the linear regression used to estimate $\Sigma(R_a)$, and the central density is deduced from $\Sigma(R_a)$ and the scale length, r_s . Again, as a fixed number of particles is used to model this inner disk, their individual masses are being changed during the run, and as in the first inner disk described, they are placed on concentric rings. The velocity profile for the inner disk is found from assuming radial hydrostatic equilibrium and using $V_o = \sqrt{GM_c/r + Ar}$, the parameter A being estimated at R_a . As in the previous disk, the radial velocity profile is determined by assuming a constant accretion rate \dot{m} throughout the inner disk.

It has been stated above for both of the inner disks that the particles are distributed in concentric rings. I now wish to describe more carefully the algorithm used to accomplish

this. In order to assure a continuous particle distribution at the boundary $r = R_a$, the inner disks are built from the outside in. The first ring radius r_1 is the radius at which the interparticle distance $(m_p/\Sigma(r_1))^{1/2}$ is equal to $2(R_a - r_1)$, and is found using the Newton-Raphson method and the appropriate density profile (equation 3.17 or 3.18). Particles placed in this and subsequent rings n , will describe the desired mass distribution subject to the condition that the mass in each ring, Δm_n , is equal to the mass in an annulus, centered on the ring radius r_n , with a width Δr_n equal to the interparticle distance. Hence, for the first ring, the mass contained within $\Delta r_1 = (m_p/\Sigma(r_1))^{1/2}$ is equal to $\Delta m_1 = m(R_a) - m(R_a - \Delta r_1)$. [In general $m(r)$ will signify the mass within radius r .] The number of particles to be placed in the ring is determined by the mass in the annulus, Δm_1 , and the particle mass, m_p . However, since the number of particles N_n placed in a ring must be an integer, and $\Delta m_n/m_p$ is in general not an integer, a small adjustment in r_n and Δr_n must be made. For the first and subsequent rings, N_n is found by rounding $\Delta m_n/m_p$ to the nearest integer value. Then the mass in the ring is redetermined: $\Delta m_n = N_n m_p$. For the first ring a new Δr_1 is redetermined by finding the radius r_M for which $m(r_M) = m(R_a) - \Delta m_1$, using the Newton-Raphson method and the appropriate expression for $m(r_M)$, and then setting $\Delta r_1 = R_a - r_M$. Finally, $r_1 = R_a - \Delta r_1/2$. For the remaining rings the same procedure is used to redetermine Δr_n from the new Δm_n , excepting that, instead of R_a , $r_L = r_{n-1} - \Delta r_{n-1}/2$ is used.

For clarity's sake, I restate the algorithm for finding the radii of the concentric rings:

1. Using the Newton-Raphson method, find r_n which satisfies the condition $2(r_L - r_n) = (m_p/\Sigma(r_n))^{1/2}$, where $r_L = R_a$ for $n = 1$, but $r_L = r_{n-1} - \Delta r_{n-1}/2$ otherwise.
2. $\Delta r_n = 2(r_L - r_n)$
3. $\Delta m_n = m(r_L) - m(r_L - \Delta r_1)$
4. Find the number of particles in ring n , N_n , by rounding $\Delta m_n/m_p$ to the nearest integer value.
5. Redetermine the mass in ring n : $\Delta m_n = N_n m_p$.
6. Using the Newton-Raphson method, determine the radius r_M for which $m(r_M) = m(r_L) - \Delta m_n$.
7. Set $\Delta r_n = r_L - r_M$.
8. Finally, $r_n = r_L - \Delta r_n/2$.
9. Proceed to next ring.

The building of the disk is discontinued when $r_M < \Delta r_n/2$.

In order to use the above algorithm, the expressions for the mass with a given radius for the two inner disks must be known. For the inner polytropic Keplerian disk:

$$m(r) = \frac{\pi G M_c}{K} (1 - k^2) \left(r - \frac{r^2}{2R_a} \right) + \pi r^2 \Sigma(R_a), \quad (3.23)$$

where k here is V/V_0 and shouldn't be confused with the index. The expression for $m(r)$ for the exponential disk is given above in equation (3.7). For the polytropic disk the above expression can be inverted to find $r(m)$, the radius within which there is mass

m . This will allow us to circumvent the use of the Newton-Raphson method in step 6) above, and instead use the expression:

$$r_M = \frac{2m_r}{B + \sqrt{B^2 + 4(\pi\Sigma(R_a) - B/2R_a)m_r}}, \quad (3.24)$$

where $B = \pi GM_c(1 - k^2)/K$, and $m_r = m(R_L) - \Delta m_k$.

CHAPTER 4

SIMULATION OF TWO-DIMENSIONAL DISKS

Parameters

To gain insight into the general behaviour and evolution of protostellar disks, a suite of models of accretion disks were simulated. Three physical parameters were varied: the central object to disk mass ratio, M_c/M_D , the ratio of specific heats, γ , and the artificial viscosity parameter α . As I am particularly interested in the early stages of the evolution of protostellar disks when the disk is self-gravitating, values of M_c/M_D of 3 and 1 were chosen.

The second physical parameter that is adjusted, γ , affects the hydrodynamical character of the gas. In particular, as it appears as an exponent in the polytropic equation of state, γ determines the compressibility of the gas. While in three dimensions a value of $\gamma \leq 4/3$ is unstable to gravitational collapse, in two dimensions $\gamma \leq 3/2$ is unstable to collapse. This is one example of how gravity is more effective in two dimensions than in three. Values of $\gamma = 2$ and $5/3$ were chosen for modeling; $\gamma = 2$ corresponds to a gas with two degrees of freedom, while $\gamma = 5/3$ to a gas with three degrees of freedom. The value $\gamma = 2$ was chosen because previous theoretical and numerical work on two-dimensional disks have used this value (Papaloizou and Savonije, 1991). However, though convenient for solving the hydrodynamical equations, $\gamma = 2$ results in a stiff equation of state. Therefore $\gamma = 5/3$ was also used in the simulations. This value of γ allows the gas to be more compressible, though it is still stable against gravitational collapse.

Of particular interest in this study is the effect of viscosity on the evolution of self-gravitating accretion disks, and the parameter used to vary the effective viscosity in SPH is the artificial viscosity parameter α . Values of $\alpha = 1, 0.5, 0.25$ were chosen, which were shown in Chapter 2 to correspond roughly to effective shear viscosity parameter values of $\alpha_\nu = .1, .05,$ and $.02$. On the basis of time scale arguments applied to real astrophysical disks, α_ν is expected to be less than 0.1, where values between 0.04 and 0.002 are most commonly argued for. As mentioned in Chapter 1, these values are much greater than those from molecular viscosity alone. Due to the as yet unresolved question of what mechanism effects this anomalous viscosity, the value of α_ν represents the greatest unknown in the physics of accretion disks.

Varying the three parameters M_c/M_D , γ , and α as described above gives twelve possible combinations, all of which are modeled. Table 1 summarizes the models discussed in this chapter, as well as the time at which each simulation was terminated, in dynamical times (see Chapter 3 for definition of units). All of the $M_c/M_D = 1$ disks terminate due to computational errors associated with adjusting the smoothing length during the neighbor searching phase of a time step, and the finite size of the arrays in which the lists of neighbors are stored. That is, the program has difficulty in adjusting the smoothing length so that particles have more than ten, but more than sixty, neighbors. This problem arises in disks in which an extreme density gradient borders a region where the surface density, and hence also the number density of particles, is very low. Such conditions arise in the disks with a mass ratio of one. In contrast, most of the $M_c/M_D = 3$ models ran to the specified time without difficulty. Their simulations

were terminated after they had run for a time comparable with the $M_c/M_D = 1$ disks. Model A3 is the sole model with a mass ratio of three that ended due to the type of difficulty mentioned above. In any event, the lengths of all the simulations were sufficient to observe important properties of accretion disks, which are noted below and in the following chapter; continuing the simulations much further, after a significant number of particles already have been removed due to accretion, would be of questionable value.

Table 4-1: Models

Model:	M_c/M_D	γ	α	Inner Disk	T_f
A1	3	2	0.25	Keplerian	33.10
A2			0.5	"	32.11
A3			1	"	33.01
A4	3	2	0.5	Exponential	24.01
B1	1	2	0.25	Exponential	38.81
B2			0.5		38.51
B3			1		31.01
C1	3	5/3	0.25	Exponential	32.61
C2			0.5		29.97
C3			1		29.55
D1	1	5/3	0.25	Exponential	33.01
D2			0.5		39.00
D3			1		32.63

While the model parameters discussed above describe the physical state of the initial conditions, another set of parameters which may affect the simulations is that of the code parameters, which determine details of the numerical simulation method. These parameters are listed in Table 2-1, with their nominal values. Ideally, if the values

chosen for the code parameters are sufficient to yield accurate simulations, the resulting simulations will be insensitive to variations in the code parameters. To test whether the simulations are accurate, several test models have been numerically evolved, each with the value of a single code parameter changed from its nominal value, so as to yield a more numerically accurate simulation. If the nominal value is sufficient, then the test simulation will show little or no change from the model run with nominal code parameters. A number of such tests are done with model D2 for the parameters θ , \mathcal{C} , and N . The comparison between the tests and model D2 are given at the end of this chapter.

The gravitational smoothing parameter, ε , is not varied for testing, because its range is restricted by requiring that it be smaller than the accretion radius, R_a , and that it must be larger than the local inter-particle separation. This last condition must be satisfied if N particles are to behave as a system with a much greater number of particles, or even a continuous medium ($N \rightarrow \infty$), as is the case here. Because ε is of single value, independent of the local number density, the latter condition cannot be satisfied in the outer, diffuse, portions of the disks. The local inter-particle separation in two dimensions is $n^{-1/2}$, with n representing the local number density. An average inter-particle distance is defined as $\bar{n}^{-1/2} = \sqrt{\pi R_D / N}$, which is equal to 0.025 for $R_D = 1$ and $N = 5131$. The condition $\varepsilon > \bar{n}^{-1/2}$ is satisfied with $\varepsilon = 0.027$, but $\varepsilon < n^{-1/2}$ for $r > 0.587$ when $N = 5131$ and $r_s = 0.25$. When N is increased to 10093 particles the condition fails for $r > 0.757$.

The other restriction, that $2\varepsilon < R_a$, is imposed so that all particles experience a Keplerian, unsoftened, gravitational potential from the central object. Therefore

increasing ε is possible if R_d is also increased. However, this is not desired, as the inner region around the central particle, which is not hydrodynamically modeled, should be a small percentage of the disk.

Evolution of Disks

General Features

Since the evolution of the models is terminated at different times, it is more useful to compare the evolution of the models at a time that all the models attain. In Table 4-2 are tabulated the total accreted mass, the central object to disk mass ratio, and the angular momentum (AM) that has been transported beyond the initial outer radius of the disk, after twenty eight dynamical times. The units of mass and angular momentum are the dimensionless units described in the previous chapter.

To qualitatively show both the effect of the viscosity parameter α and the ratio of specific heats γ , the models are shown after ten dynamical times in Figures 4-1 and 4-3, and after thirty dynamical times in Figures 4-2 and 4-4. For both the $M_c/M_D = 1$ and 3 models, increasing the viscosity effectively damps the amplitude of the modes. This is even more noticeable at early times, where viscosity also suppresses the growth of the modes. Of particular of interest is that the viscosity has suppressed the $m=1$ mode in the models having $M_c/M_D = 1$. The mode is weakly visible in model B3 in Figure 4-4, but not visible in model D3, and a dominating $m=1$ mode remains absent in this last model until the simulation is terminated at $T=32$. This later model represents the one exception in the modal evolution of the $M_c/M_D = 1$ disks, which otherwise become dominated by a slowly growing $m=1$ mode. This mode is of a tidal nature, resulting

from the gravitational interaction between the massive central object and the disk, and has been predicted by others (Savonije *et al.*, 1992).

Table 4-2: Models at T=28 dynamical times

Models	α	Accreted		Transported
		mass (10^{-2})	M_c/M_D	AM (10^{-2})
A1	0.25	3.504	3.790	2.501
A2	0.5	2.851	3.645	2.923
A3	1.0	2.042	3.477	2.562
B1	0.25	15.26	1.985	10.68
B2	0.5	14.23	1.895	9.877
B3	1.0	11.69	1.697	11.67
C1	0.25	2.856	3.646	2.844
C2	0.5	1.808	3.430	2.808
C3	1.0	2.125	3.493	2.440
D1	0.25	16.34	2.084	14.15
D2	0.5	13.47	1.833	9.461
D3	1.0	7.466	1.422	8.323

The effects of varying the ratio of specific heats are more subtle. At early times, when the modes are growing, the models with smaller γ show slightly more density contrast between arm and inter-arm regions. At the later times, such as those shown in the Figures 4-3 and 4-4, this trend is not apparent in the $\alpha = 0.25$ and 0.5 disks, and is reversed in the $\alpha = 1$ disks. Another effect that is more obvious in the $\alpha = 0.25$ and 0.5 disks is that those with $\gamma = 5/3$ tend to show greater density variations along the arms of the non-axisymmetric modes. At times this density variability will develop into clumping in the outer portions of the arms, such as that seen in model D1.

A more detailed description of the evolution of the modes, and of the mass accretion observed in these disks are given in the next two sections. In the $M_c/M_D = 1$ disks another distinctive feature is seen, and that is the formation of small satellites in the outer portions of disks. This is discussed in the following chapter.

Modal Evolution

The evolution of the simulated disks is typified by the presence of many transient modes. In none of the $M_c/M_D = 3$ disks does a single mode dominate the disk for more than a few dynamical times. In all disks the growth of the non-axisymmetric modes follows the general pattern that modes with higher mode number show faster growth rates. However, as the amplitude in all modes increase, the modes of higher mode number will peak and then decrease, while the modes of lower mode number continue to grow. Hence, as the amplitude of the density fluctuations increases during the early part of the evolution, different modes achieve temporary dominance. The maximum amplitude is attained usually by an $m=2$ or $m=3$ mode. When the maximum power in a mode has diminished it will usually not remain suppressed, but will later increase again to become the mode with maximum power. In other words, the modes will initially grow at different rates, but eventually the amplitude of all the modes seem to fluctuate at about the same amplitude. The evolution of the maximum power of one model is shown in Figure 2-6.

The same behavior is initially seen in the $M_c/M_D = 1$ disks, but a slowly growing $m=1$ mode eventually dominates most of these disks, with one exception. Unlike the other models, the dominating $m=1$ mode is not a spiral wave, but instead is the result of the central object and disk rotating about their center of mass. That is, the “central”

object, representing the star, is no longer centered on the mass distribution of the system. Another characteristic of the modal evolution of these disks is that an $m=2$ mode will dominate before this $m=1$ mode becomes the dominant feature. Figure 4-5 shows a plot of the maximum power reached by modes 1 through 4, between a radius of 0.1 and 1.0, in the simulation of model B2.

It should be noted that characterizing the evolution by the maximum power present in each mode may have limitations. Because the Fourier analysis was done on the normalized density, the outer portions of the disk will primarily be represented, for this part of the disk possesses the greatest density contrasts.

This transient behavior, as well as the number of modes present, does not allow the frequencies to be unambiguously identified with a particular mode. The presence of many initial transient modes is not surprising, as many modes will be excited by the perturbation that has been seeded into the initial conditions. However, the persistence of this transient behavior in a dissipative system is unexpected. In addition, the details of the evolution of the modes for any given disk seem to be sensitive to initial conditions, in the sense that a change in a computational parameter, such as the tolerance angle or the number of particles, will cause the disk's evolution to diverge in detail from a corresponding model that is otherwise the same. This chaotic behavior brings into question the practicality of describing in detail the complex evolution of the modes that are observed, though general features of the evolution have been noted. (This same sensitivity to initial conditions is seen in meteorology, where dissipation also is present.)

This behavior can be explained in two ways: the fluctuations are either reflecting the physical behavior of such a system, or they are due to the failure of the numerical method. If the behavior is physical then two processes may be occurring either separately, or in conjunction. One process is the interaction between modes, where energy is being transferred between modes continuously. In the disks that are modeled here, interaction between the modes is enhanced because of the proximity of the corotation of a mode m with the outer Lindblad resonance of the $m - 1$ mode, and the inner Lindblad resonance of the $m+1$ mode, in the frequency – radius domain. In Figure 4-6 the location of the resonances are shown. In such a system energy is easily transferred from one mode to another. The other possible process is that the power in a mode is fluctuating due to the presence of two modes with different frequencies, but the same mode number, which would result in the oscillation of the amplitude of that mode.

Alternatively, the cause of the fluctuation in power may be due to a failure of the SPH method. An example would be if the method could not describe a shock front properly once it had developed a large amplitude. In this case the post-shock oscillations could destroy a wave mode that has developed a shock front. There may be additional problems in the outer portions of the disks where the number density of SPH particles can become too low to describe regions of low density. It is in this part of the disk that the highest normalized density amplitudes appear. Gaps in the particle distribution appear between the arms of the non-axisymmetric modes; these gaps are low density regions that are under-represented. To see if the transient character of the modes resulted from such problems, models were run with 10093 particles. This increased the resolution of

the method, as the smoothing length h is inversely proportional to the square root of the number density. While the under-represented regions of the disk were further out in the disk than in the models with 5130 particles, the same transient behavior of the modes was observed, though differing in detail. This suggests that the transient behaviour of modes is an inherent effect found in such disks, rather than being a numerical effect.

Accretion

The rate of accretion onto the central object is determined by the rate of angular momentum transport effected in the disk by shear viscosity and the non-axisymmetric modes. In most of the models simulated, the mass accretion rate typically begins at a large value, and within 2π dynamical times approaches a constant rate, which is more sensitive to the viscosity parameter than to the ratio of specific heats. In some cases a constant accretion rate is immediately realized (models A3 and C3). The mass accretion is shown in Figures 4-7 through 4-11 for all models. The constant accretion rates in the models were measured by fitting a line to the appropriate section of the mass accretion curves, as shown by one example in the latter figure. Not surprisingly, the models with $M_c/M_D = 1$ show higher constant accretion rates than the $M_c/M_D = 3$ disks. Although it is not clear why constant accretion rates should be an endemic feature of accretion disks, they are seen to some degree in most of the simulated disks, with only models A1 and C1 showing a small continual decrease of their accretion rate with time. For these two cases the accretion rates measured are the asymptotic accretion rate for most of the simulations.

Interestingly, though a constant accretion rate is seen in almost all models, it is not necessarily maintained throughout the run; in some cases the accretion rates change to new constant values, sometimes with almost discontinuous abruptness. Also worthy of note is the insensitivity of the initial accretion rates of models D1 and D2 to viscosity ($\alpha = 0.25$ and 0.5), as the mass accretion of these models is nearly equal. It is after new constant accretion rates are re-established that the rates of these two models diverge. This behavior is also observed in models C1 and C2, and therefore seems to be characteristic of the $M_c/M_D = 1$ models with lower values of α . With the exception of model D3, all of the $M_c/M_D = 1$ models show at least one accretion rate shift, as well as the $M_c/M_D = 3$ models A3 and C3, which both have a viscosity parameter equal to one. Two models show a second accretion rate change: models A3 and D1. For the $M_c/M_D = 1$ models, the first accretion rate change takes place at a transition point in the modal evolution of the disk. The maximum power in the $m=2$ mode peaks preceding the change in accretion rate, and the $m=1$ mode becomes the dominant mode in the disk. The second accretion rate change seen in model D1 takes place when the power in all modes with $m > 1$ abruptly falls to 0.2, one third the value of the $m=1$ mode.

The exception to this behavior seen in the $M_c/M_D = 1$ models is model D3, which does not develop a dominant $m=1$ mode. In this model the accretion rate changes very slowly until a constant accretion rate is established. Unlike the $M_c/M_D = 1$ models, the constant accretion rate changes observed in models A3 and C3 are not clearly matched with identifiable events in their modal evolution. In Table 3 the accretion rates, and the times that they are attained, are given for all of the models in dimensionless units. To

convert the accretion rates to physical units, multiply the rates by the conversion factor (M_T/T_D), the total mass divided by the dynamical time in the desired units. For instance, using $M_T = 1M_\odot$ and the dynamical time $T_D = 159$ years, the accretion rates are found to range between $2.36 \times 10^{-6}M_\odot/\text{yr}$ and $4.31 \times 10^{-5}M_\odot/\text{yr}$.

Table 4-3: Accretion Rates ($\times 10^{-3}$)

Model	\dot{m}_1	t_1	\dot{m}_2	t_2	\dot{m}_3	t_3
A1	1.00	NA				
A2	0.813	8				
A3	1.13	0	0.688	8	0.375	20
A4	0.670	10				
B1	3.88	7	6.11	18		
B2	4.00	7	5.00	20		
B3	3.00	9	7.32	22.5		
C1	0.813	NA				
C2	0.531	7				
C3	0.875	0	0.375	20.5		
D1	4.25	6	6.86	18	4.63	27
D2	4.51	4	3.05	26		
D3	2.13	17				

More surprising is the dependence of the mass accretion on viscosity, for the mass accretion increases as α decreases. This stated dependence is evident in Figures 4-7 through 4-10, though it is not established in the $M_c/M_D = 1$ disks with lower viscosity parameter values until the constant accretion rate changes. The inverse dependence of mass accretion upon shear viscosity is reflected in the mass accretion rates, which are plotted in Figures 4-12 and 4-13. For the $M_c/M_D = 3$ models the trend is established

after the secondary accretion rates have been attained in the $\alpha = 1$ disks, which are the rates that coincide temporally with the mass accretion rates observed in the models with smaller viscosity parameters. It is also the secondary accretion rates in the $M_c/M_D = 1$ models that show the same trend with shear viscosity, with the exception of the secondary accretion rate of model B3.

This counter intuitive relationship between viscosity and mass accretion occurs due to the combined actions of the global non-axisymmetric modes and local viscous processes, which are not independent of one another. When both are present the viscosity damps the more efficient mechanism of angular momentum transport provided by the non-axisymmetric modes. The numerical experiments show that lower effective shear viscosity allows the non-axisymmetric modes to attain greater strength, which in turn become more effective at transporting angular momentum. This damping action of the viscosity is not only qualitatively apparent, but is also evidenced by the rms amplitudes of the density fluctuations.

In all the models, the total rms amplitude of the normalized density initially increases exponentially with time. Most models then undergo a period of linear growth until a saturation level is reached by the model; model D3 attains a saturation level immediately after the initial exponential growth. Two models, D1 and D3, also show a second saturation level being attained later in the simulation. In addition to this general behavior of the normalized amplitudes, erratic fluctuations in its magnitude are apparent that take place on a time scale of about two dynamical times. The fluctuations grow in amplitude until the saturation level is reached, and are not obviously present until the period of

linear growth begins. The shear viscosity suppresses the rms amplitude of the density fluctuations, causing lower saturation levels to be attained. Also, the viscosity generally enhances the rate of the initial exponential growth.

Table 4—4: Normalized density amplitude growth rates and saturation levels.

Model	Viscosity (α)	Growth Rate (at $R = 0.55$)	Growth Rate (at $R = 1$)	Saturation level(s)
A1	0.25	0.7	1.7	0.7
A2	0.5	1.3	1.8	0.55
A3	1	1.7	2.1	0.5
A4	0.5	1.3	1.8	0.5
B1	0.25	1.1	1.5	1.1
B2	0.5	0.9	1.5	0.9
B3	1	1.4	2.1	0.8
C1	0.25	0.7	2.0	0.9
C2	0.5	0.9	2.1	0.8
C3	1	1.5	2.6	Not attained (<0.6)
D1	0.25	1.0	1.8	0.65
D2	0.5	1.0	2.2	1.2 (0.7)
D3	1	1.2	3.0	0.3 (0.45)

The growth rates and saturation levels of the rms amplitude of the normalized density fluctuations also vary with radius. The saturation level of the rms amplitude increases with radius, as do the growth rates. Table 4 shows the growth rates for the various models at the radial distances of 0.55 and 1.0 from the center of mass of the system. The saturation level given corresponds to the smaller radius of 0.55.

The method of treating the central region and accretion in the hydrodynamic code was intended to let the properties of the disk drive accretion, rather than be determined by the imposed inner boundary condition around the central object. To test the degree to which I am successful, models were run with two different inner disks (models A2 and A4), and another with a larger R_a (model A5). Both types of inner disks, an exponential disk and a massless Keplerian disk, provide pressure support at the inner boundary by maintaining a continuous surface density and velocity field across the boundary (see Chapter 3 for further details). Models A2 and A4 are identical except for the treatment of the inner region, and the mass accretion in the two models is nearly the same (see Figure 4-11). However, since the massless Keplerian disk results in the density gradient being discontinuous at the boundary, the inner exponential disk was preferred. To what degree the size of R_a alone influences accretion is tested by model A5, which is identical to model A4 excepting the size of the inner region. The value of R_a for model A5, 0.0635, deviates from the nominal value of 0.0544. As can be seen in Figure 4-11, the size of the inner accretion region does not significantly affect the mass accretion.

The initialization of an encounter model is described in chapter 3, which includes an encountering particle with a mass of 0.5, and a disk that is otherwise equivalent to model D2. In the resulting simulation it is found that the encounter takes place approximately after ten dynamical times. The time sequence of this encounter is shown in the following chapter, in Figures 5-9 through 5-12. Beside robbing a significant fraction of mass from the disk, the encountering particle also greatly enhances the mass accretion onto the central object. This is shown in Figure 4-14.

Tests of Code Parameters

As discussed at the beginning of this chapter, models with differing values of code parameters, but equivalent model parameters, should be compared to confirm that the choice of code parameters yield consistent results. Since it was impractical to run such a series of tests for all combinations of the physical parameters, a single model was chosen: model D2. The test models are designated T1, T2, and T3. In model T1 the tolerance angle, θ , is given a value of 0.5, resulting in a gravitational approximation closer to a direct summation of the gravitational terms due to N particles. In model T2 the Courant parameter, \mathcal{C} , is equal to 0.2, resulting in smaller integration time steps. In model T3 the number of particles used in the simulation is increased to 10093 particles. The mass accretion of these three test models are shown with the mass accretion of model D2 in Figure 4-15.

Model T1, with a smaller tolerance angle than that of D2, is nearly equivalent to model D2, indicating that the gravitational forces are being calculated to a sufficient degree of accuracy with $\theta = 0.7$. Model T2 initially has the same mass accretion as D2, but then deviates after twenty dynamical times. This may indicate that the hydrodynamic calculations require smaller time steps to yield sufficiently accurate results.

Model T3, which possesses 10093 particles, differs most dramatically from model D2. However, these two models are not *physically* equivalent for several reasons. By increasing the number of particles by nearly a factor of two the local smoothing length is decreased by approximately a factor of $2^{-1/2}$. This effectively changes the local viscosity of the disk. An additional difference in model D2 and T3 is also seen when the modal

evolution of model T3 is inspected: a $m=1$ mode does not eventually dominate the disk of model T3 as it does model D2. This doubtless accounts for the smaller mass accretion rate in model T3 in spite of the fact that, with a smaller smoothing lengths, the effective local shear viscosity is smaller than in model D2. As mentioned above, weaker shear viscosity typically leads to larger accretion rates. The reason that model T3 does not develop a dominating $m=1$ mode is most likely due to the fact that its initial density perturbation also differs from that of model D2. This is because the density perturbation was initially seeded into the disk by displacing the particles from the positions required to describe a smooth exponential disk. These displacements are random in direction and Gaussian in magnitude, with a standard deviation that is a fraction of the local interparticle spacing. Because model T3 has a smaller interparticle spacing than model D2 the initial noise in density in the two models are not the same. Hence, as models T3 and D2 are not equivalent physically, these two models can not be directly compared.

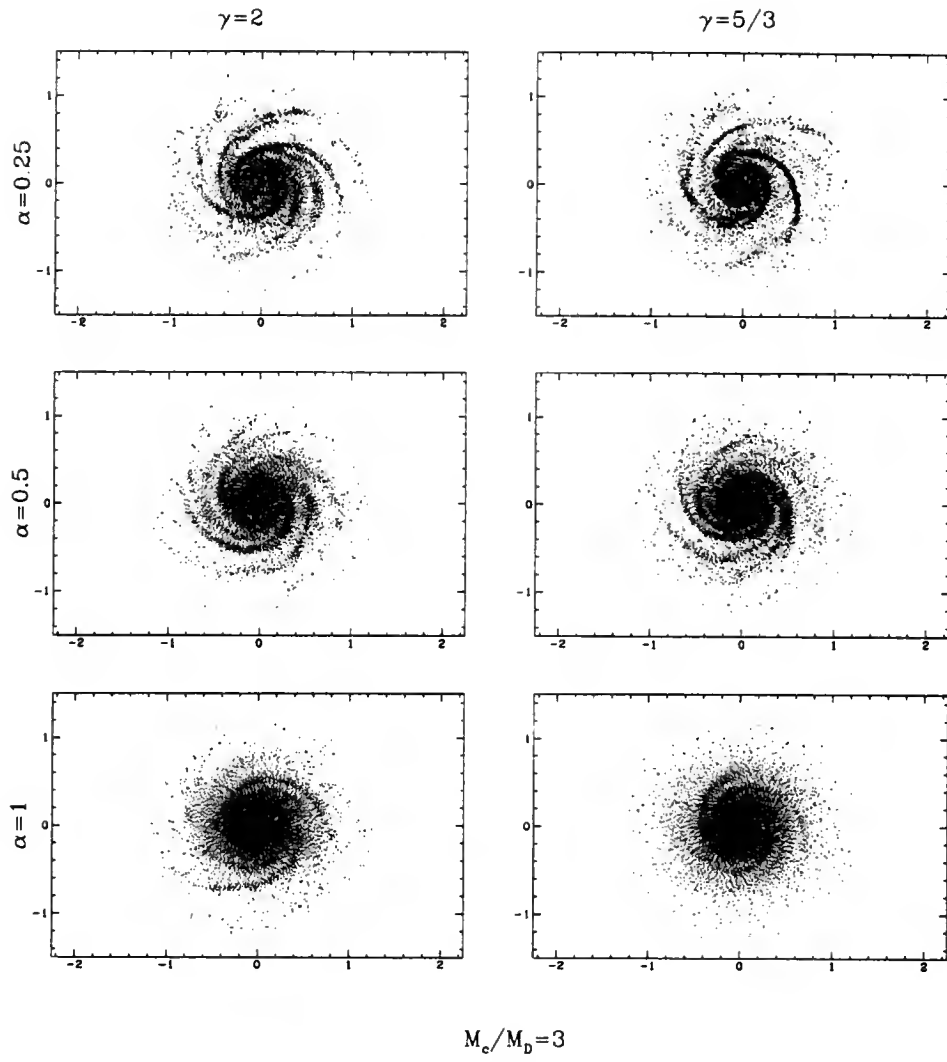


Figure 4-1: Particle distribution of $M_c/M_D = 3$ models at Time = 10.0 dynamical times.

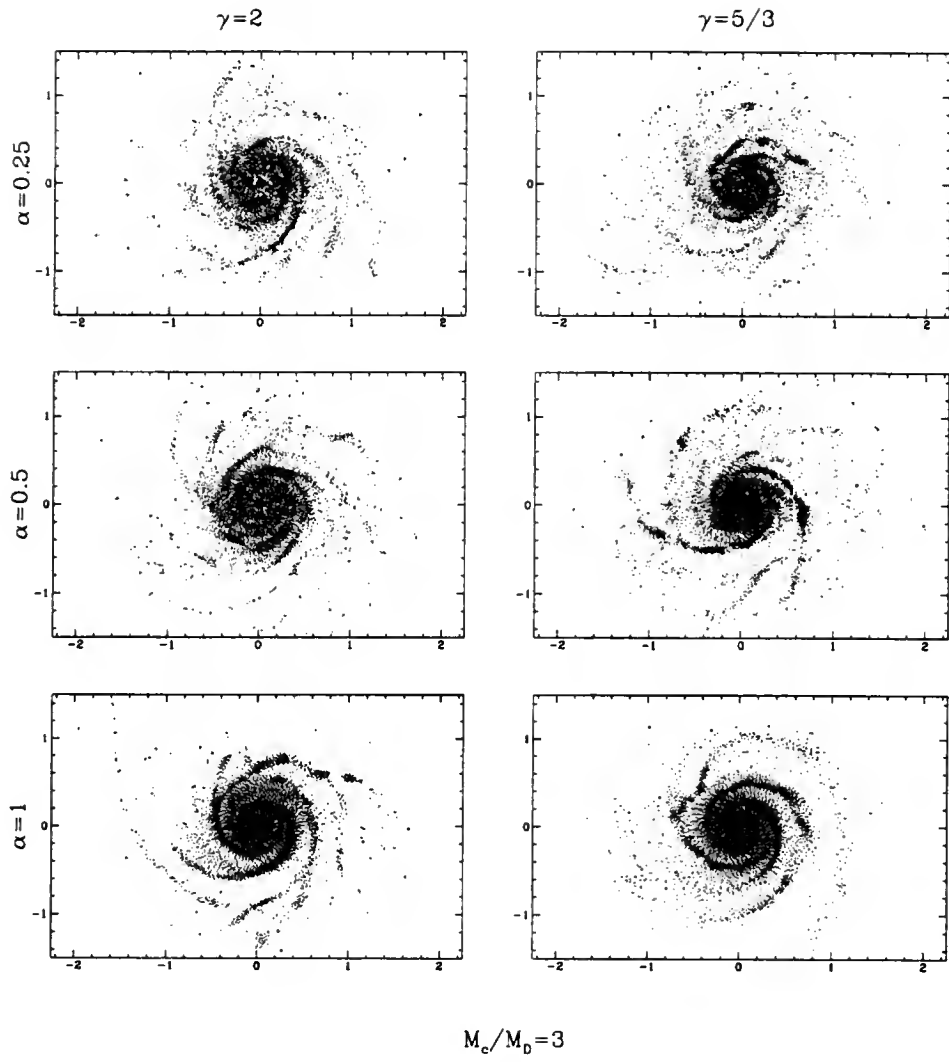


Figure 4-2: Particle distribution of $M_c/M_D = 3$ models at Time = 29.5 dynamical times.

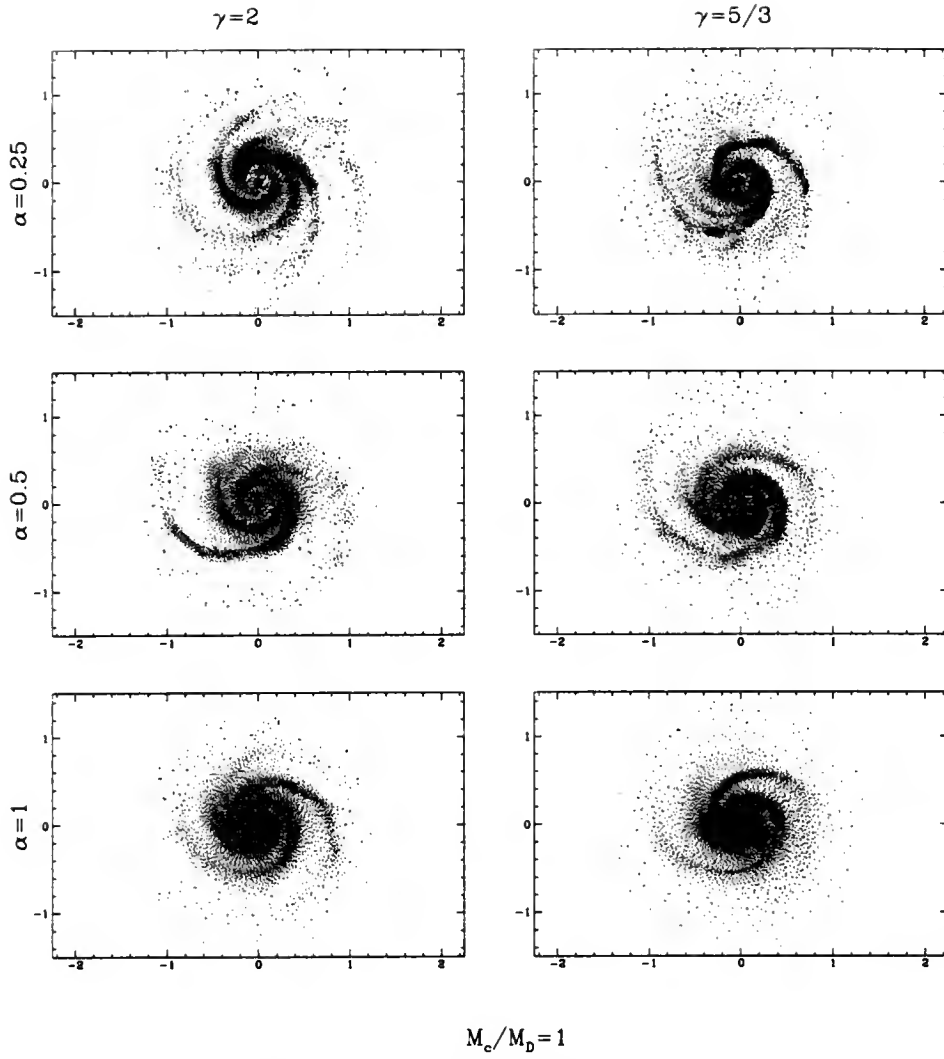


Figure 4-3: Particle distribution of $M_c/M_D = 1$ models at Time = 10.0 dynamical times.

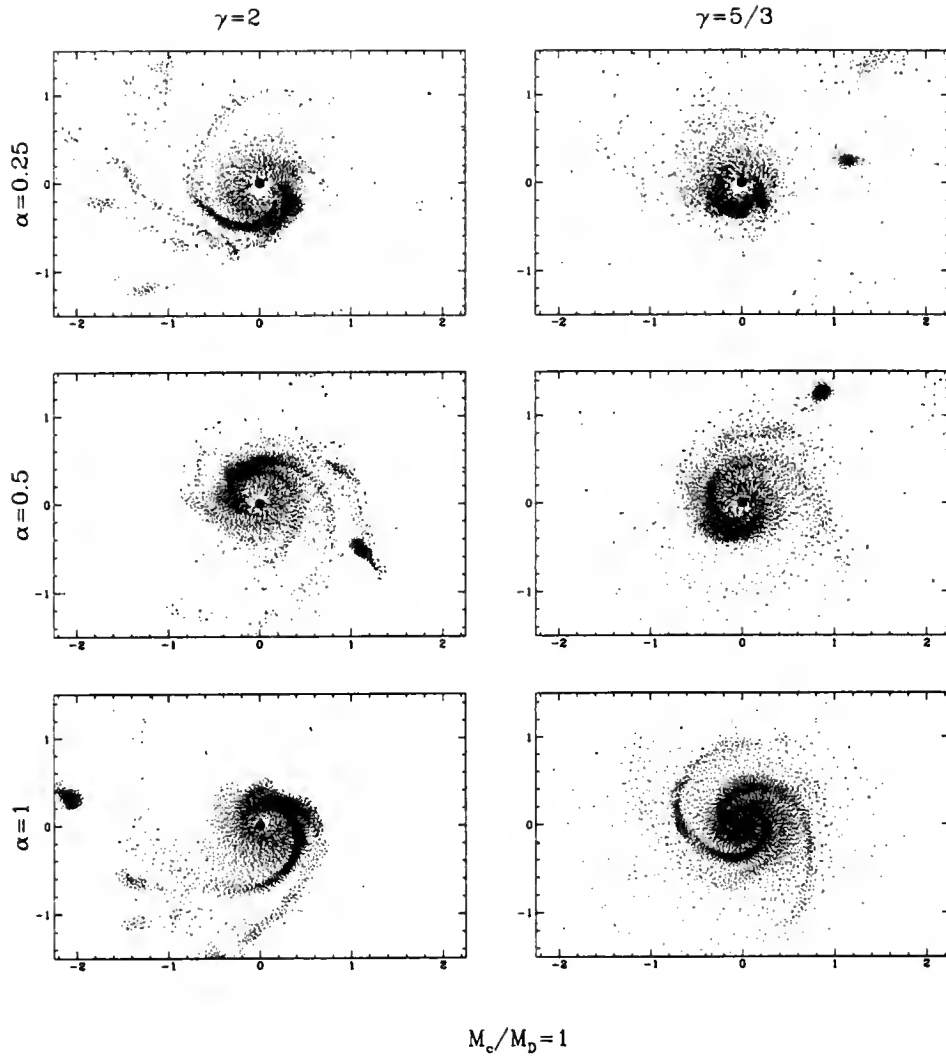


Figure 4-4: Particle distribution of $M_c/M_D = 1$ models at Time = 30.0 dynamical times.

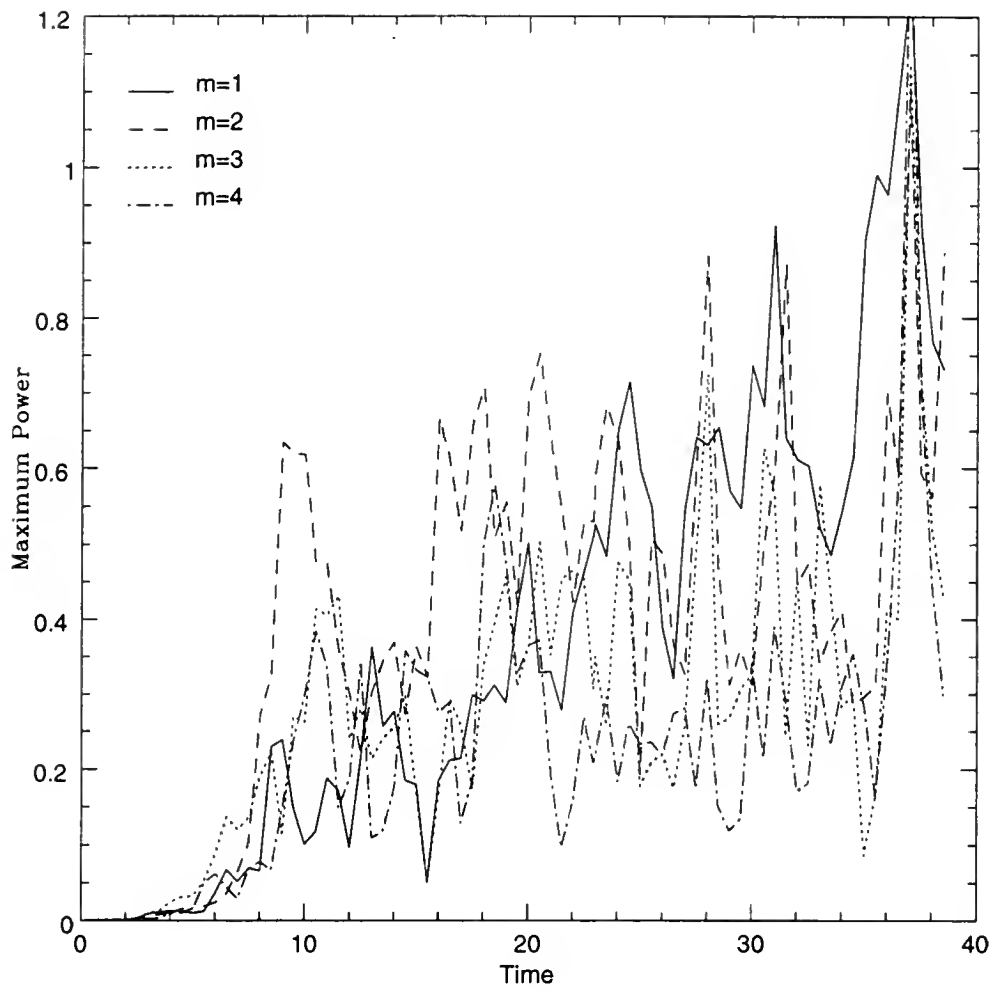


Figure 4-5: Maximum power in modes 1 through 4 for model B2.

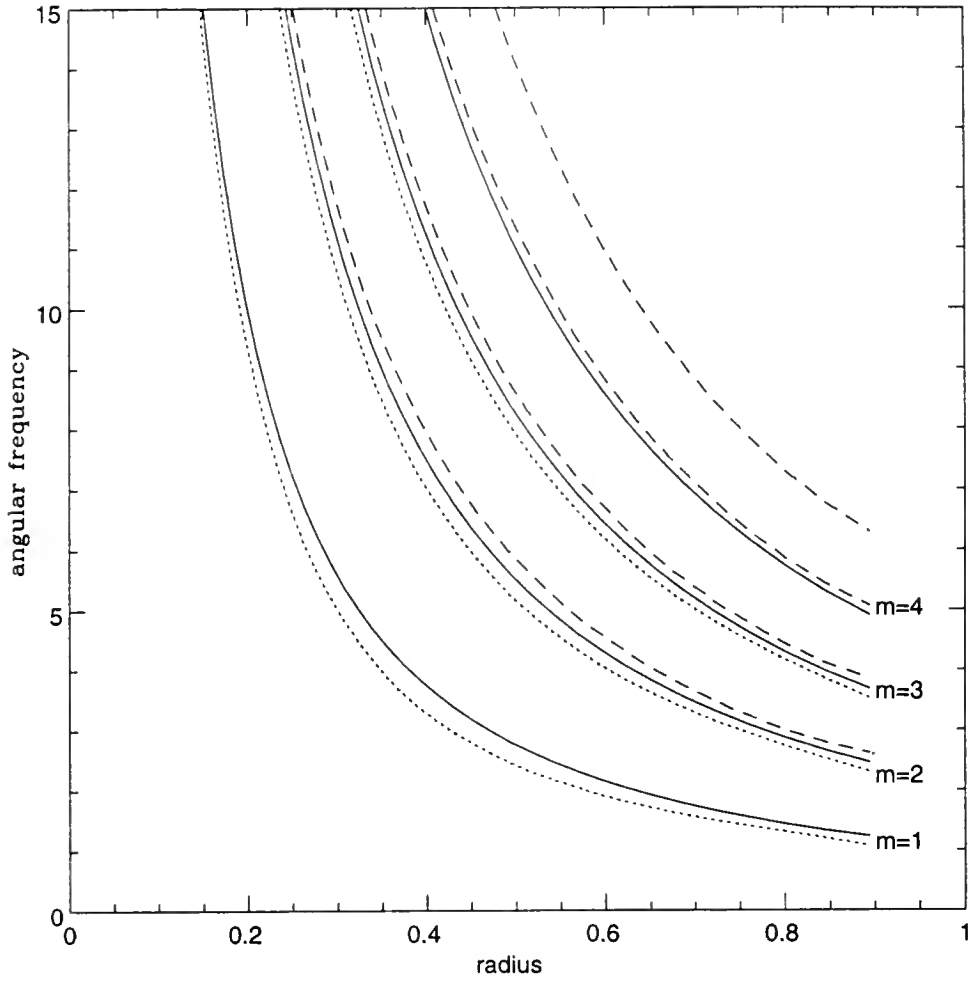


Figure 4-6: Resonances in an accretion disk with $M_c/M_D = 3$. Solid lines correspond to the corotation resonance, the dotted lines to the Inner Lindblad resonance, and the dashed lines to the Outer Lindblad resonance.

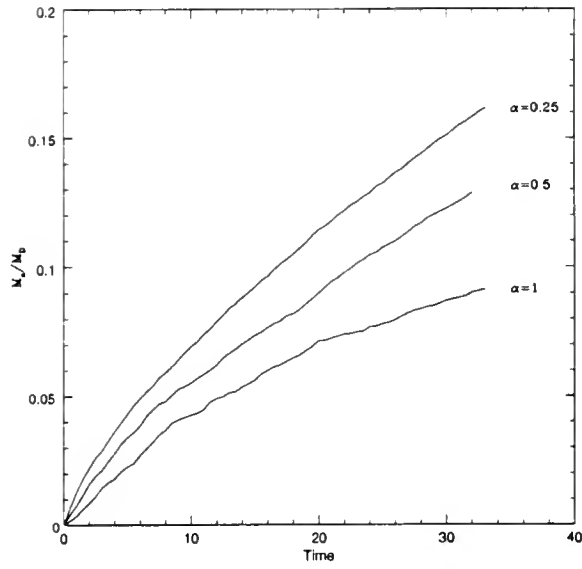


Figure 4-7: Mass accretion, as a fraction of initial disk mass, for models A1 through A3.

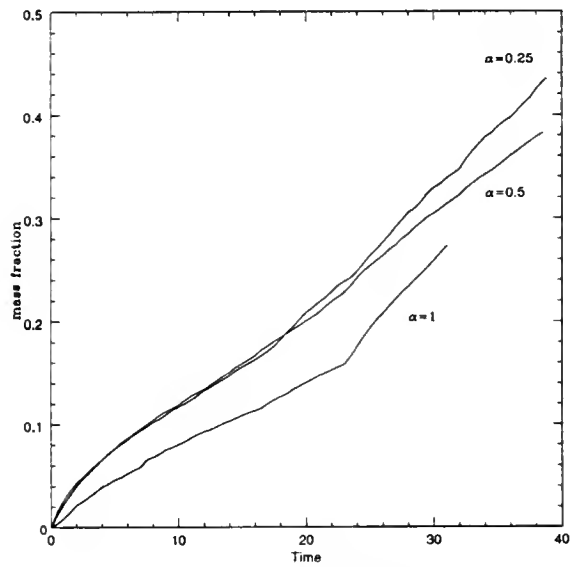


Figure 4-8: Mass accretion, as a fraction of initial disk mass, for B models.

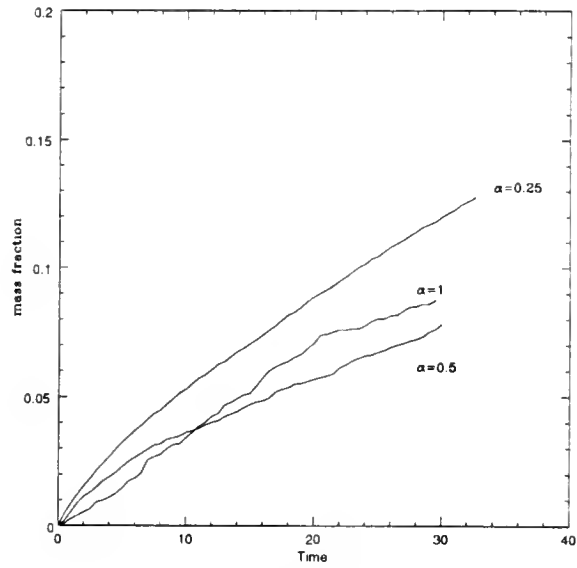


Figure 4-9: Mass accretion, as a fraction of initial disk mass, for C models.

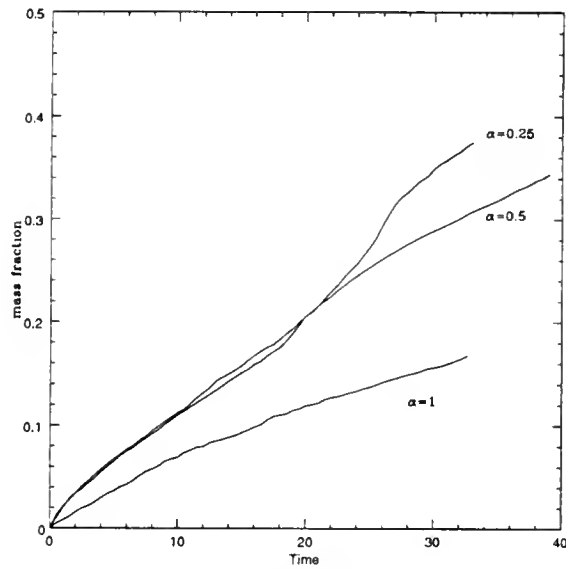


Figure 4-10: Mass accretion, as a fraction of initial disk mass, for D models.

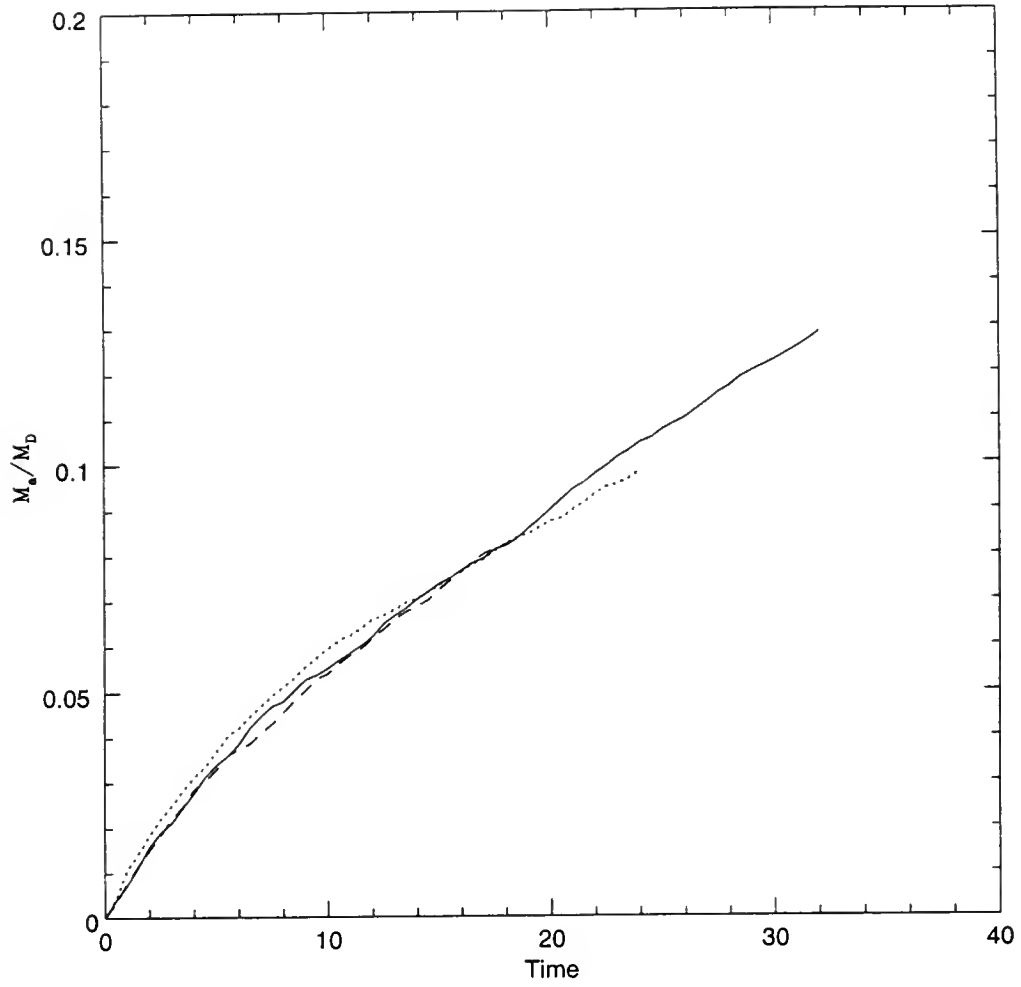


Figure 4-11: Mass accretion in models A2, A4, and A5, as a fraction of the initial disk mass.

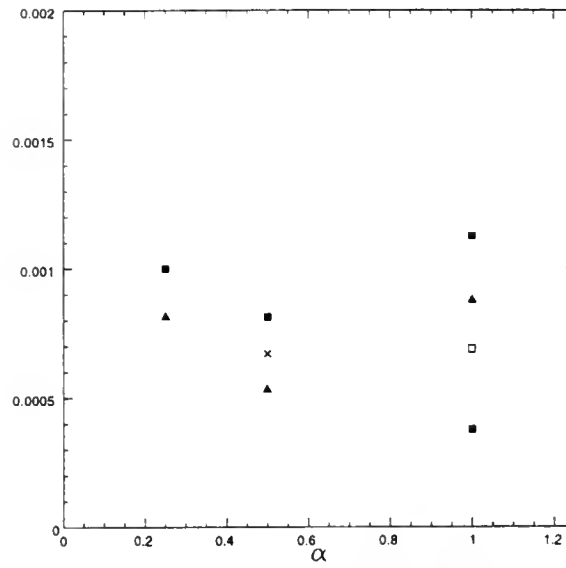


Figure 4-12: Constant mass accretion rates for $M_c/M_D = 3$ accretion disks.

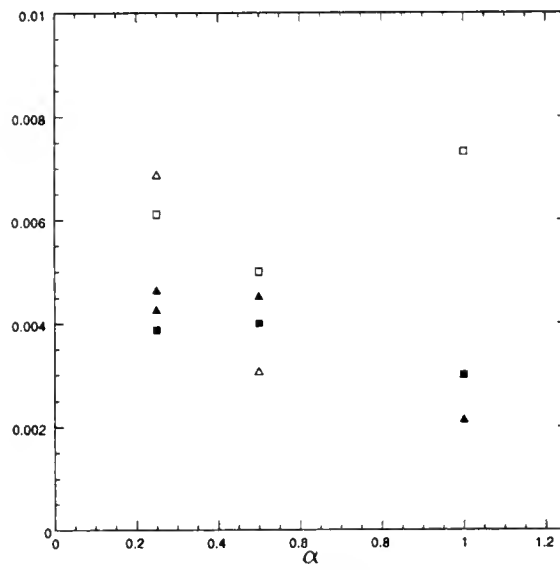


Figure 4-13: Constant mass accretion rates for $M_c/M_D = 1$ accretion disks.

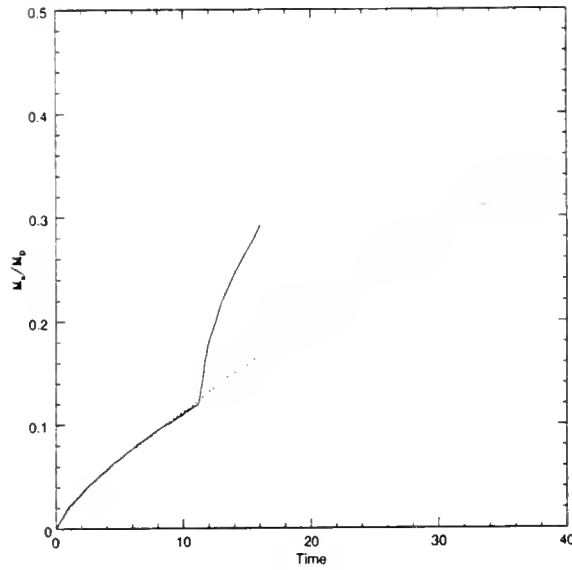


Figure 4-14: Mass accretion of the encounter model (solid line) compared to the mass accretion of model D2 (dotted line).

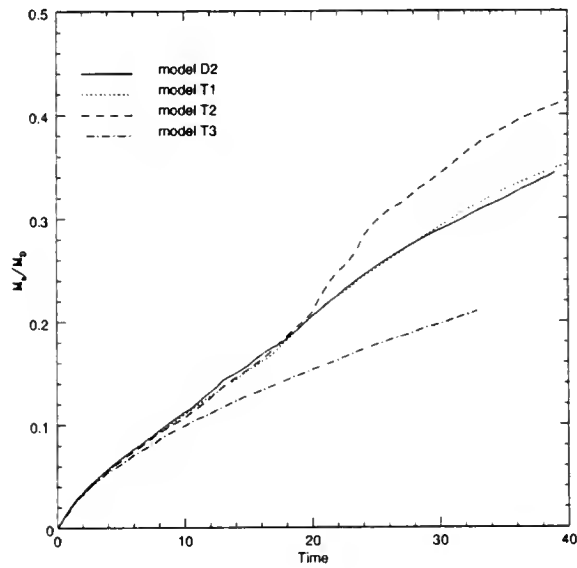


Figure 4-15: Mass accretion, as a fraction of initial disk mass, for model D2 is shown with three test models.

CHAPTER 5

FORMATION OF SATELLITES

The greatest difference between disks with higher and lower star to disk mass ratios is that all disks that develop a dominating $m=1$ mode also form satellites. This includes all disks with $M_c/M_D = 1$, with the exception of model D3. By way of example, the later evolutionary sequence of model D2 is shown in Figure 5-1 and Figure 5-2. Figure 5-3 shows the final configuration of the disk, with the path of the satellite shown since its formation. The satellite forms from a clump of gas that is recognizable as a distinct structure at time = 25. At this time it is part of a spiral arm of a $m=1$ mode that extends to large radii. The mass in the initial clump of gas is $0.013M_T$, and at time = 39 the satellite has a mass of $0.028M_T$. Also, as the mass of the satellite increases, and the combined mass of the star and the disk interior to the companion decreases, their mean separation decreases. A closer view of the satellite is shown in Figure 5-4, which also shows its radial density profile.

Other models form multiple satellites, which can interact with one another. Table 2 provides an overview of the formation and evolution of the satellites, giving the initial separation R_i between the mass and the central object – disk center of mass, the initial mass M_i , the time of formation T_i , the final separation R_f , and the final mass of the satellite, M_f . The final time, T_f , corresponds to either the end of the simulation, or to when the satellite was destroyed.

Table 5-1: Satellites

Model	Satellite	R_i	M_i ($\times 10^{-2}$)	T_i	R_f	M_f ($\times 10^{-2}$)	T_f
B1	1	1.96	0.448	31.5	3.25	0.536	38.8
	2	1.57	0.409	22.5	4.72	0.692	38.8
	3	2.01	0.370	23.0	4.57	0.419	38.8
B2	1	2.23	0.546	24.0	3.72	1.277	38.5
	2	1.68	0.975	24.0	0.81	4.045	37.5*
B3	1	1.33	1.374	21.0	1.48	4.776	31.0
	2	1.41	0.760	24.0	2.26	1.72	31.0
D1	1	1.21	0.770	18.5	2.31	1.920	33.0
	2	1.17	1.150	17.5	0.98	2.174	33.0
	3	1.30	0.741	22.0	1.65	0.955	33.0
	4	1.17	1.053	19.0	0.67	1.803	24.5*
D2	1	1.50	1.257	25.0	1.24	2.797	39.0

* Indicated satellites are reabsorbed into the disk.

Orbital Evolution

In the absence of close encounters with other satellites, the orbital evolution can be understood by considering the simpler system of two point masses in circular motion about a common center of mass. In such a two body system the separation of the two masses is determined by the masses and the orbital angular momentum, given explicitly by

$$a = \frac{L^2}{G} \frac{m}{(m_1 m_2)^2}, \quad (5.1)$$

where L is the orbital angular momentum about the center of mass, and $m = m_1 + m_2$ is the total mass. The above equation shows how the equivalent separation between

two bodies can decrease, assuming they remain on circular orbits. If the total mass and orbital angular momentum remain constant, and mass is transferred from the more massive body (the primary), m_1 , to its less massive companion (the secondary), m_2 , then the separation will decrease as $1/(m_1 m_2)^2$. The separation will also decrease if the orbital angular momentum decreases. A change in the orbital angular momentum can occur by a variety of mechanisms: a) external torques on the system, such as an outer satellite, b) internal torques, which can transfer orbital angular momentum into, or out of, the spin angular momentum associated with an extended asymmetrical mass distribution about one of the bodies, c) mass loss from the system, which will carry away orbital angular momentum with it, and conversely, d) mass accretion onto the system. While mass loss from the system decreases the orbital angular momentum, it may cause the separation to increase. From the above equation, and assuming that the mass loss process does not alter the velocities of the two masses, and that the secondary's mass remains constant, the condition for *increasing* the separation with mass loss from the primary is $m_1/m_2 > \frac{1}{2}(1 + \sqrt{17}) \simeq 2.56$.

To understand the orbital evolution of the formed satellites, the above model is applied as an approximation. The mass of each of the satellites, and the mass within the minimum separation between the satellite and central object, is found. These masses are taken to be the masses of the primary and secondary respectively. Hence, the primary mass m_1 consists of the central object and the mass in the disk, as well as any inner satellites, that lie within the minimum separation. In addition, the orbital angular momentum associated with these two masses, with respect to their center of mass, is found. Using the above

formula (equation 5.1), the equivalent separation, a , of a system in circular motion, with the same orbital angular momentum and masses, was found. Often the formed satellites exhibit substantial eccentricity, which results in the actual separation oscillating about the equivalent separation. All of the above mechanisms for altering the separation between the disk-star and the satellite are observed to some degree. Mass is commonly lost from the disk as mass is transported outward with angular momentum. As the primary mass of the disk-star far exceeds the mass of the satellite, this mass loss results in the separation increasing, if the mass of the satellite remains constant. Interaction with the non-axisymmetric modes of the disk, which act as a reservoir of angular momentum, is also important. Though this model is an approximation it gives sufficient insight to understand the orbital evolution of the satellites.

In the case of model D2, where only a single satellite is present, the decrease in the separation is primarily due to a change in the mass ratio of the primary and the satellite. In model B2 (Figure 5-5) two satellites form almost simultaneously at the same polar angle. The outer satellite then spirals outward, while the inner satellite spirals inward. Initially this can be understood as resulting from the interaction of the two satellites. However, while the outer satellite's orbital angular momentum rises sharply soon after formation, a comparable drop in the inner satellite's orbital angular momentum is not observed; the orbital angular momentum of the inner satellite initially remains constant. Therefore any angular momentum transferred from the inner to the outer satellite must be offset by an equal transfer of angular momentum from the disk to the inner satellite, or else angular momentum is being transferred from the disk directly to the outer satellite. Later the

angular momentum of the inner satellite increases dramatically, due to interaction with the non-axisymmetric modes in the disk. However, this increase of the inner satellite's orbital angular momentum is not enough to counter the effects of mass transfer from the disk to the satellite, and it continues to spiral inward to eventually be reabsorbed into the disk. At the same time the outer satellite continues to slowly gain orbital angular momentum, even while its own mass, and the mass interior to it remains constant. The source of this orbital angular momentum must be from the disk and satellite interior to it, as no external torques are present.

In model B3 two satellites also form (Figure 5-6). However, due to a different configuration of the satellites, they do not strongly interact. The innermost satellite remains at nearly a constant separation for four dynamical times, then gently increases to its maximum peak value at $T=28.5$, and then diminishes to its final value. Though the separation does not vary greatly, the mass and orbital angular momentum show abrupt increases, making "steps" to higher values. These steps are correlated with collisions of the satellite with arms of a mode that extends out to the orbital radius of the satellite and overtakes it. Between these collisions the separation between the satellite and primary continues to decrease due to mass loss from the primary. In contrast to the inner satellite, the outer satellite's separation grows continuously after its formation. Its mass and orbital angular momentum also remain constant for four dynamical times, after which both increase as it encounters material moving outward from the disk. The continual change in the separation is again due to the decrease of the primary's mass.

In model B1 three satellites form (Figure 5-7), and all move outward to greater radii. Soon after their formation, the three satellites' masses remain constant. The orbital angular momentum of the first two satellites increase nearly linearly with time during the simulation, while the outermost satellite's orbital angular momentum remains nearly constant during its trajectory. Again, the increase in separation is due largely to the primary mass, that corresponds to each satellite, decreasing in magnitude. However, the net increase in orbital angular momentum in this system of bodies, *despite the loss of mass to the primary*, is clear evidence of angular momentum being transferred from the disk, which has a strong $m=1$ mode, to the satellites.

Model D1, which I have left for last, is the most complicated, as four satellites are formed (Figure 5-8). The first two satellites are clearly self-gravitating and distinct structures. A third is much more diffuse, but remains as a recognizably distinct structure for eleven dynamical times, perhaps with the aid of the first two satellites, which are at larger radii. The fourth satellite is short lived, encountering the first satellite soon after its own formation, and is consequently reabsorbed into the disk. Until just previous to this collision the first satellite's mass and angular momentum are constant with time, but both increase abruptly during the collision. Afterward, the orbital angular momentum of the first satellite increases linearly with time, presumably interacting further with the fourth satellite, as well as with the $m=1$ mode of the disk. The mass of the second satellite remains constant with time soon after its formation, while the orbital angular momentum initially increases, then levels, and gradually decreases. The separation also increases, then decreases. This is due not only to ellipticity but also the loss of orbital

angular momentum, as the equivalent separation of a system with circular rotation also decreases. Obviously the change in orbital angular momentum is due to interaction with other components in this system, though it is not clear what the interactions are. The orbital evolution of the third satellite is the simplest: its mass, orbital angular momentum, and equivalent separation all remain constant, while the real separation fluctuates about the equivalent separation due to the ellipticity of its orbit.

Formation Conditions

So far I have only described the orbital evolution of the satellites after their formation, but have not described the formation of any of the satellites. In general, from a simple visual inspection of the models, several formation processes seem to be present. One is the gradual accumulation of diffuse material well outside the denser, inner, portions of the disk. This process may be encouraged by previously formed satellites at larger radii. Examples of this are satellites B1-1, B1-2, and D1-3. The second process involves outer arm fragments that have become detached from the rest of the arm of a non-axisymmetric mode extending into the diffuse region around the disk. After becoming detached most of these fragments will dissipate, be reabsorbed by the disk, or encounter a satellite. However a few will gradually accumulate mass and contract, or collide with another fragment, to form a self-gravitating clump. This formation process seems to be responsible for most of the satellites. The third process is very like the second, where clumping within an arm of a mode occurs until the self-gravitating satellite forms and continues on its own trajectory, separating from the arm in which it was born in. This is the case for the first satellite discussed, the one observed to form in model D2. Once the

satellites have formed they do not continue to gravitationally collapse, but reach a near equilibrium because of the form of the equation of state; a polytropic equation of state, with a ratio of specific heats greater than 1.5, is stable against gravitational collapse.

The identification of the satellites was done by mere inspection at later times, and each traced back in time for as long as it could be visually identified as distinct from its surroundings. To test whether the satellites are indeed self-gravitating, I compared their mass with the Jeans Mass, which is the minimum mass necessary, within a specified area, to be self-gravitating. That the satellites are self-gravitating is already born out by their remaining coherent structures for many dynamical times. However, I wished to identify at what time each satellite initially becomes a self-gravitating entity. An expression for the Jeans Mass appropriate for two dimensions is derived below from the two-dimensional virial theorem, whose derivation immediately follows.

In two dimensions the radial component of the hydrodynamic equation of motion is

$$\left(\frac{\partial}{\partial t} + V_r \frac{\partial}{\partial r} \right) V_r = -\frac{1}{\Sigma} \frac{\partial p}{\partial r} - \frac{\partial \varphi}{\partial r}, \quad (5.2)$$

with the assumption of axisymmetry. In the Lagrangian formulation we can rewrite the above equation as

$$\Sigma \frac{\partial^2 r}{\partial t^2} = -\frac{\partial p}{\partial r} - \Sigma \frac{\partial \varphi}{\partial r}. \quad (5.3)$$

Multiplying by r and integrating with respect to $d\tau = 2\pi r dr$ we have

$$\int \Sigma r \frac{\partial^2 r}{\partial t^2} d\tau = - \int r \frac{\partial p}{\partial r} d\tau - \int \Sigma r \frac{\partial \varphi}{\partial r} d\tau. \quad (5.4)$$

Given

$$r \frac{\partial^2 r}{\partial t^2} = \frac{1}{2} \frac{\partial^2 (r^2)}{\partial t^2} - 2 \left[\frac{1}{2} \left(\frac{\partial r}{\partial t} \right)^2 \right], \quad (5.5)$$

the left hand side of the previous equation can be rewritten as

$$\int \frac{1}{2} \frac{\partial^2 (r^2)}{\partial t^2} dm_r - 2 \int \frac{1}{2} \left(\frac{\partial r}{\partial t} \right)^2 dm_r. \quad (5.6)$$

The first integral is equal to $\frac{1}{2} \ddot{I}$, one half the second order derivative of the moment of inertia with respect to time, and the second integral is equal to $2K_B$, two times the bulk kinetic energy associated with radial oscillations.

Considering now the right hand side of equation (5.4), the first integral can be simplified by integrating by parts:

$$- \int r \frac{\partial p}{\partial r} d\tau = -2\pi \int r^2 dp = -2\pi r^2 P_s + 2\pi \int p 2r dr, \quad (5.7)$$

where P_s is the surface pressure. Also, by writing

$$2\pi \int p 2r dr = 2 \int p d\tau = 2\bar{P}A, \quad (5.8)$$

where A is the total area considered, and \bar{P} is taken as a average pressure, the first integral on the right hand side is taken as being equal to $-P_s A + 2\bar{P}A$. Taking the second integral on the right hand side of equation (5.4), and integrating by parts renders

$$- \int 2\pi \Sigma r^2 \frac{\partial \varphi}{\partial r} dr = -2\pi r^2 \Sigma \varphi + 2 \int \varphi \Sigma d\tau + \int r \frac{\partial \Sigma}{\partial r} \varphi d\tau. \quad (5.9)$$

The first term on the right hand side, the surface term, is equal to zero, while the second term is equal to four times W , the gravitational potential energy of the configuration, using $\frac{1}{2} \int \varphi \Sigma d\tau = W$ (Binney & Tremaine, 1987). Putting all the respective parts together, the full two-dimensional virial theorem is

$$\frac{1}{2} \ddot{I} = 2K_B - P_s A + 2\bar{P}A + 4W + \int_0^R r \frac{\partial \Sigma}{\partial r} \varphi d\tau. \quad (5.10)$$

The last integral is dependent upon the radial density profile of the axisymmetric configuration, a term that does not occur in the three dimensional virial theorem. Assuming that $\tilde{I} = K_B = P_s = 0$, and that $\partial\Sigma/\partial r = 0$ (a uniform density profile), the virial theorem reduces to the equation $\bar{P}A = -2W$. Taking $\bar{P} = K\bar{\Sigma}^\gamma$, $A = \pi R^2$ and $\bar{\Sigma} = M_J/A$ the expression for the Jeans Mass is found to be

$$M_J = \left[-\frac{2W}{K} A^{1/\gamma} \right]^{\gamma-1}. \quad (5.11)$$

Because this expression pertains to the simple case of a uniform density configuration, it represents a maximum estimate of the necessary mass for a self-gravitating two-dimensional mass configuration to be in virial equilibrium, under the assumption of a polytropic equation of state. A non-uniform density distribution would result smaller Jeans Mass due to a negative contribution from the integral in equation (5.10).

To calculate the Jeans Mass for a satellite the gravitational potential energy is found by taking the direct sum over the particle pairs:

$$-\frac{1}{2} \sum_j m_j \sum_i \frac{Gm_i}{r_{ij}}. \quad (5.12)$$

This expression is for unsoftened gravity, resulting in a larger estimate of the Jeans Mass than would be found if the softened gravity were used in the calculation of W . The size of the region R is found by taking the particle furthest from the center of mass of the satellite within a $.4 \times .4$ region roughly centered on the satellite. It is found that the mass of all satellites well exceeds this maximum estimate of the Jeans Mass at all times. However, the Jeans Mass may not be the relevant criterion to define whether a satellite has enough self-gravity to be considered a coherent structure. After all, the satellites

do not exist in isolation, but within the potential well of the central companion and its disk. Hence, a criterion based on the competition between self-gravity and tidal forces is more appropriate.

The criterion for the tidal disruption of a satellite, assumed to be held together by its own self-gravity alone, is found from considering the net force upon a particle at the satellite's surface. In a frame of reference rotating with the satellite, the total force on a particle at the satellite's surface is composed of the gravitational forces from the primary and the satellite, as well as the centrifugal force. By expressing both the centrifugal force, and the gravitational force from the primary body, as truncated Taylor expansions about the center of the satellite, the criterion for tidal disruption of a satellite is found to be

$$m_2 < m_1 \frac{R^3}{r^3}, \quad (5.13)$$

where R is the radius of the satellite. The radius of the satellite is taken to be twice the radius at which the surface density of the satellite is equal to the average surface density of the satellite. The assumption was also made that the primary and the satellite were spherical mass distributions, hence applying the above criterion assumes the two-dimensional satellite represents a three-dimensional spherical satellite with the same mass and radius. Again, all the identified satellites exceed the critical mass for tidal disruption, which identifies them as gravitationally coherent structures in the gravitational field of the primary, with few exceptions: a few of the satellites do not possess the necessary mass only at the earliest times that they are identified. This implies that, if the two-dimensional satellites were in three dimensions, they have been traced back to the time of their initial formation.

Encounter Model

As mentioned in the previous chapter, a single encounter model was run, and the effect of the encounter on the mass accretion onto the central object was noted. Here, in figures 5-9 through 5-12, the time sequence of the encounter is shown. Figure 5-12 shows the post-encounter configuration at time equal to 16.0 dynamical times. Note the formation of three satellites in the tidal tail.

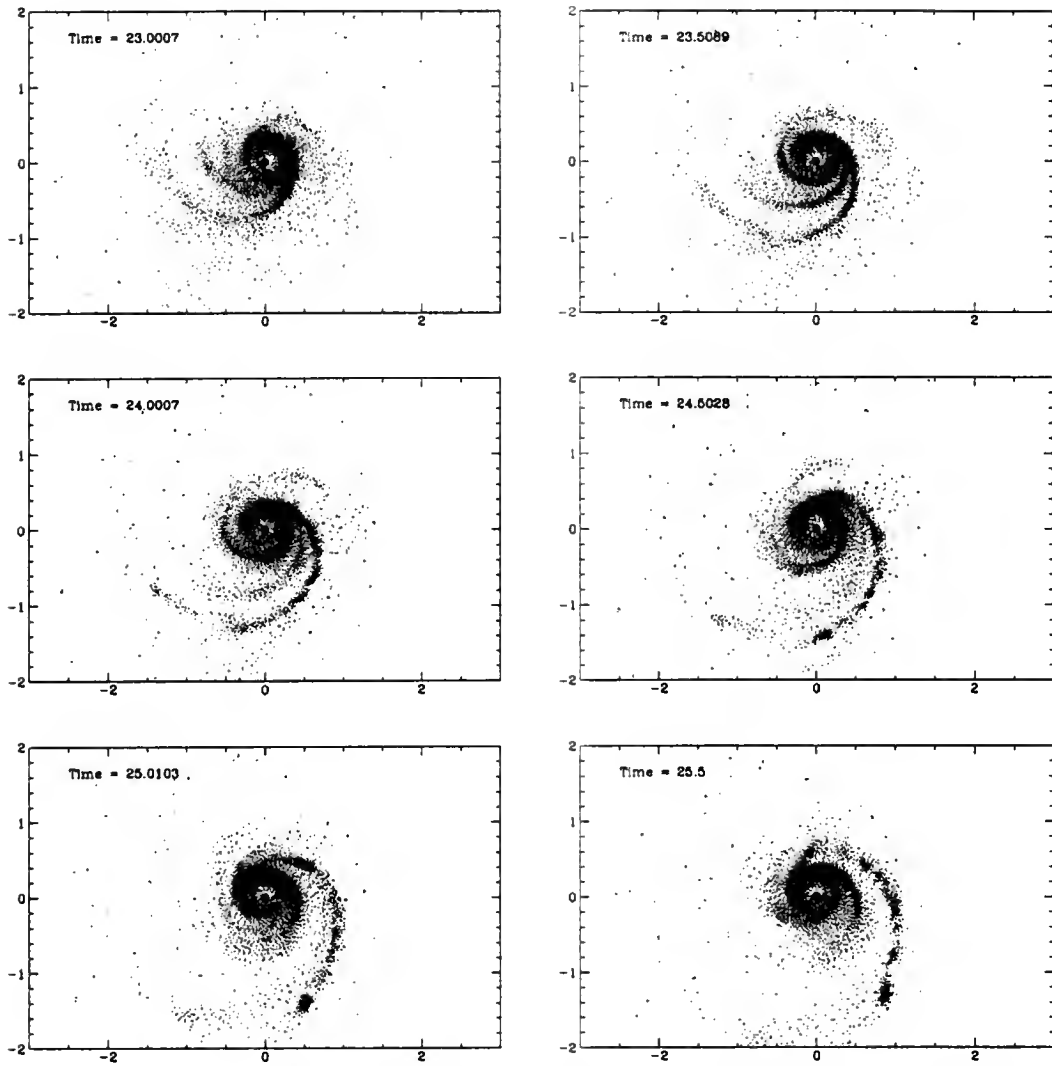


Figure 5-1: Evolutionary sequence of model D2 showing formation of satellite.

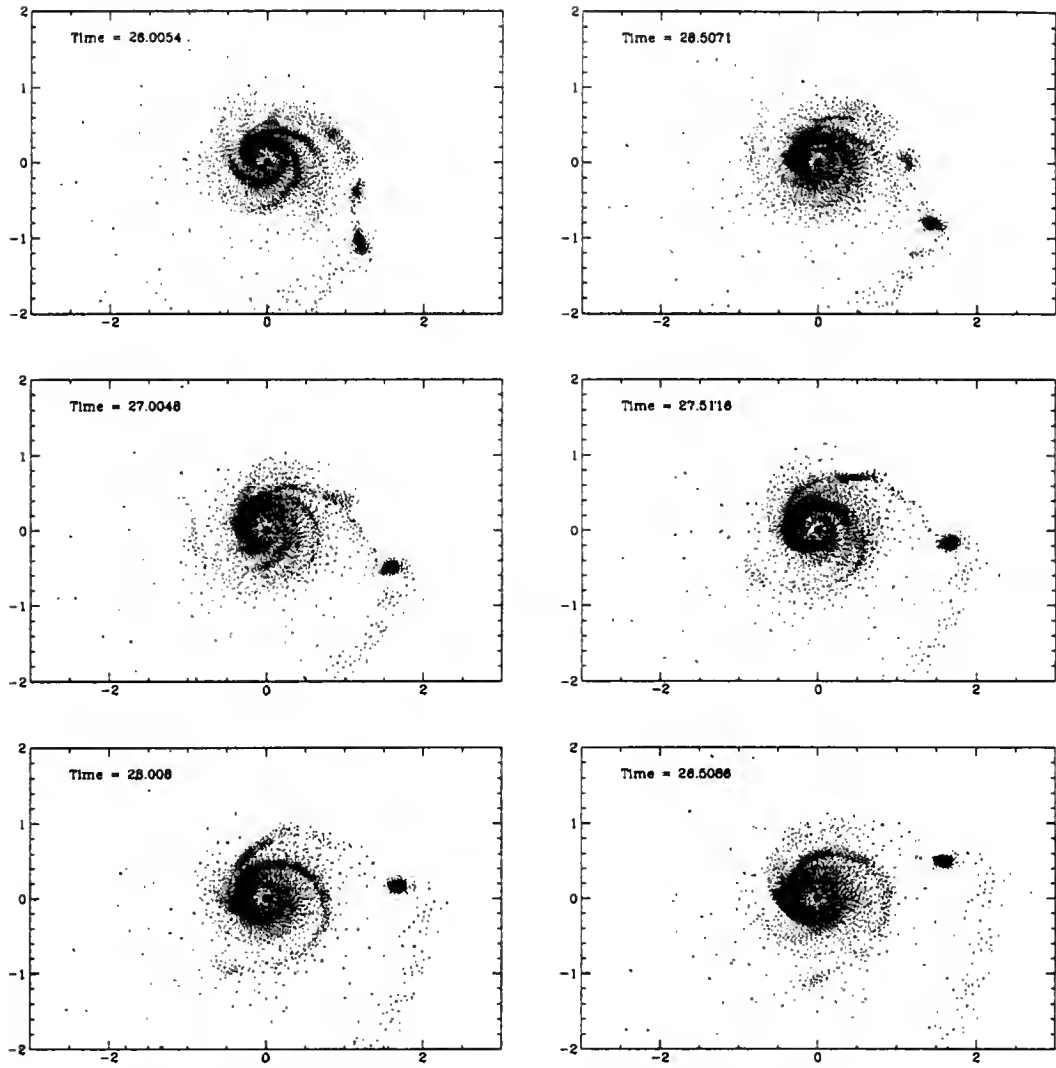


Figure 5-2: Evolutionary sequence of model D2 continued.

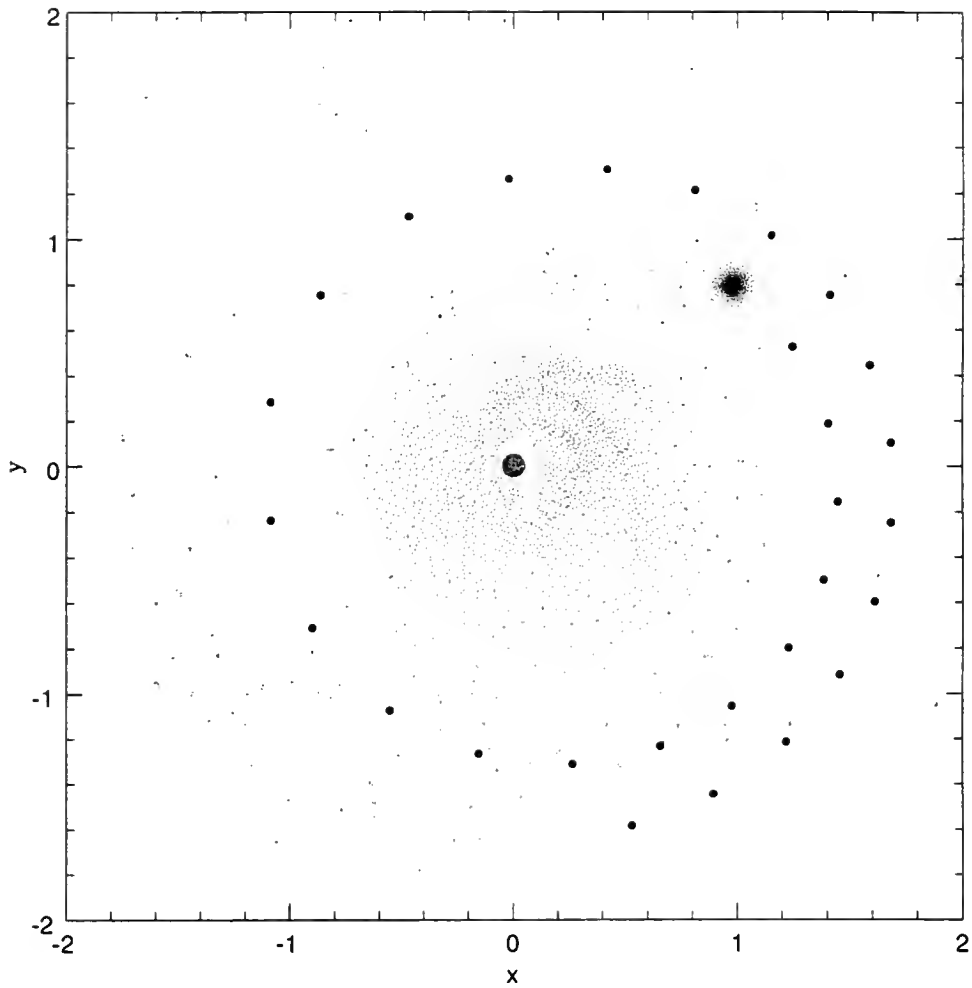


Figure 5-3: Final configuration of model D2, showing the position of the satellite at earlier times.

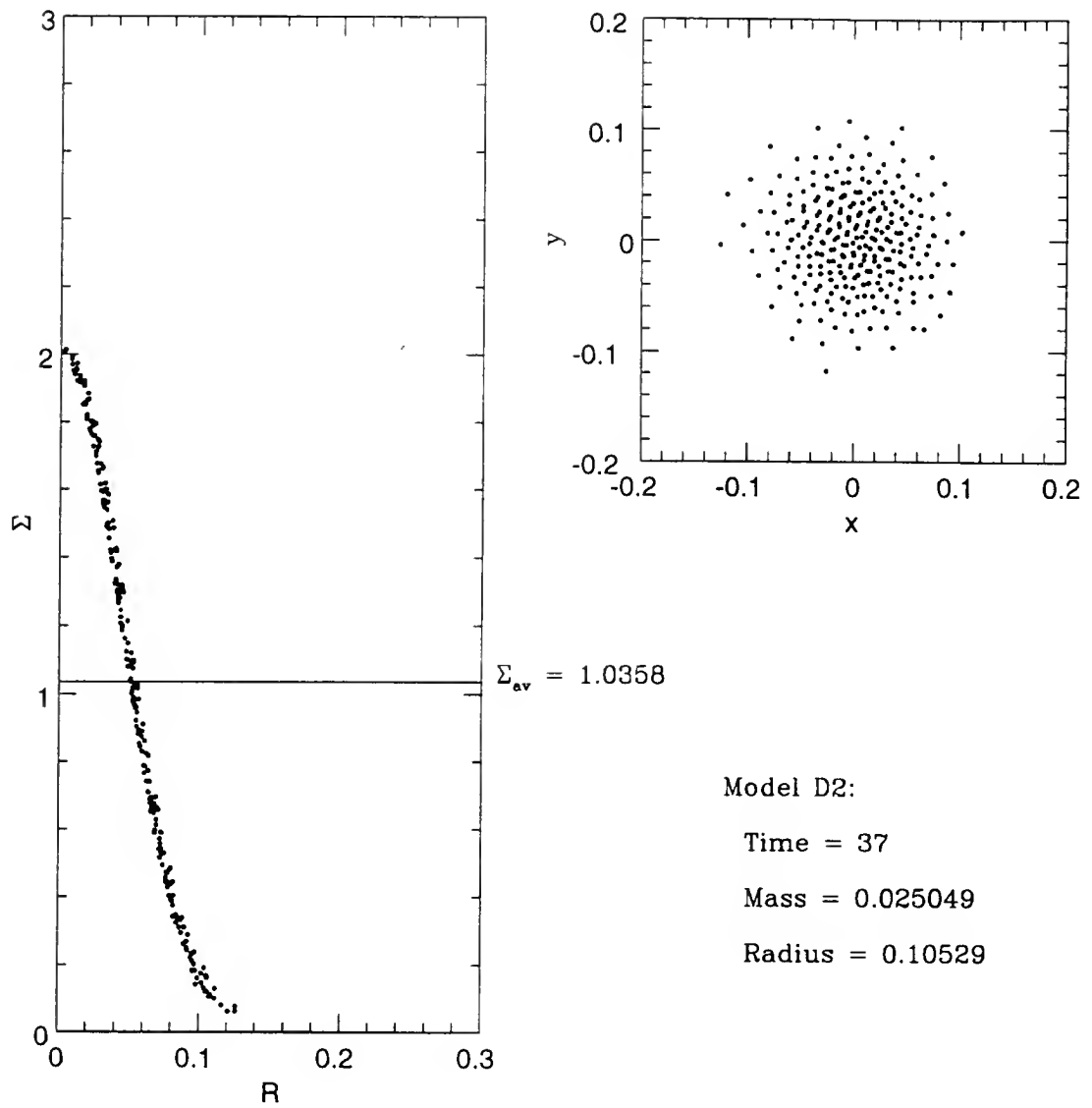


Figure 5-4: Radial density profile and particle distribution of satellite D2-1.

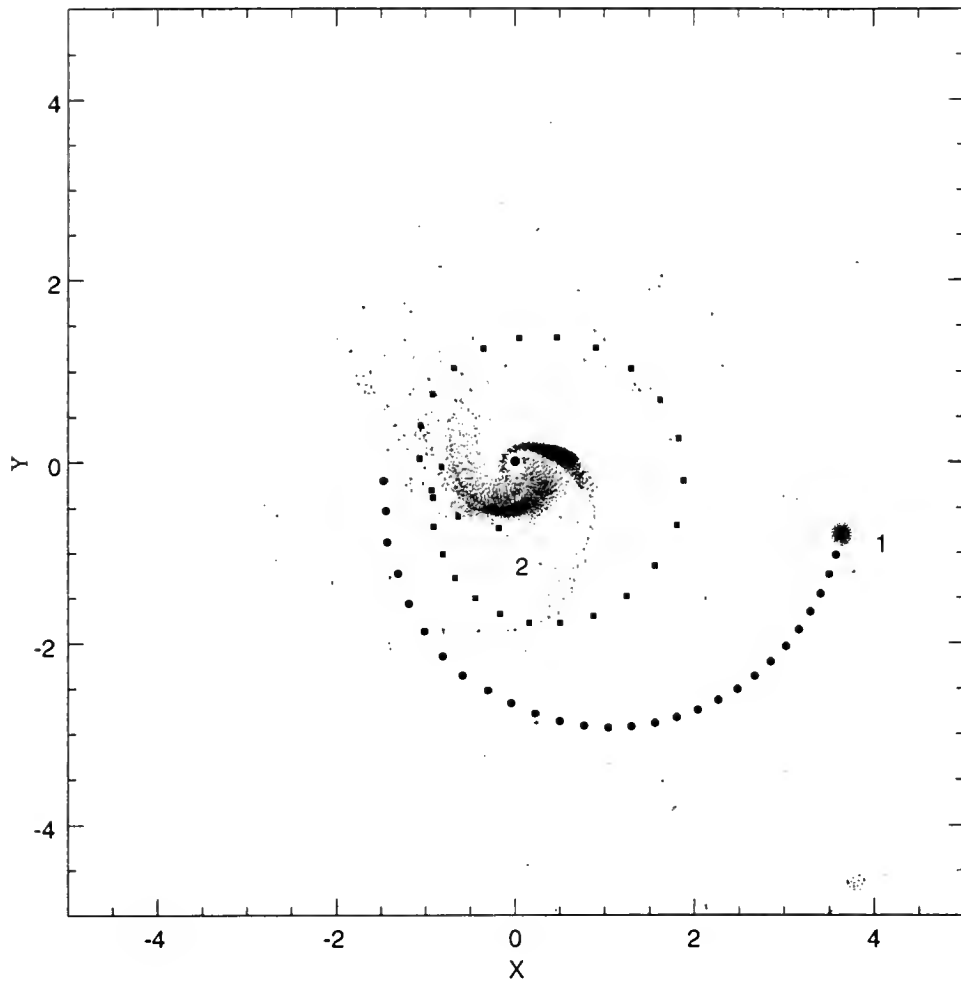


Figure 5-5: Final configuration of model B2, showing the position of the satellites at earlier times.

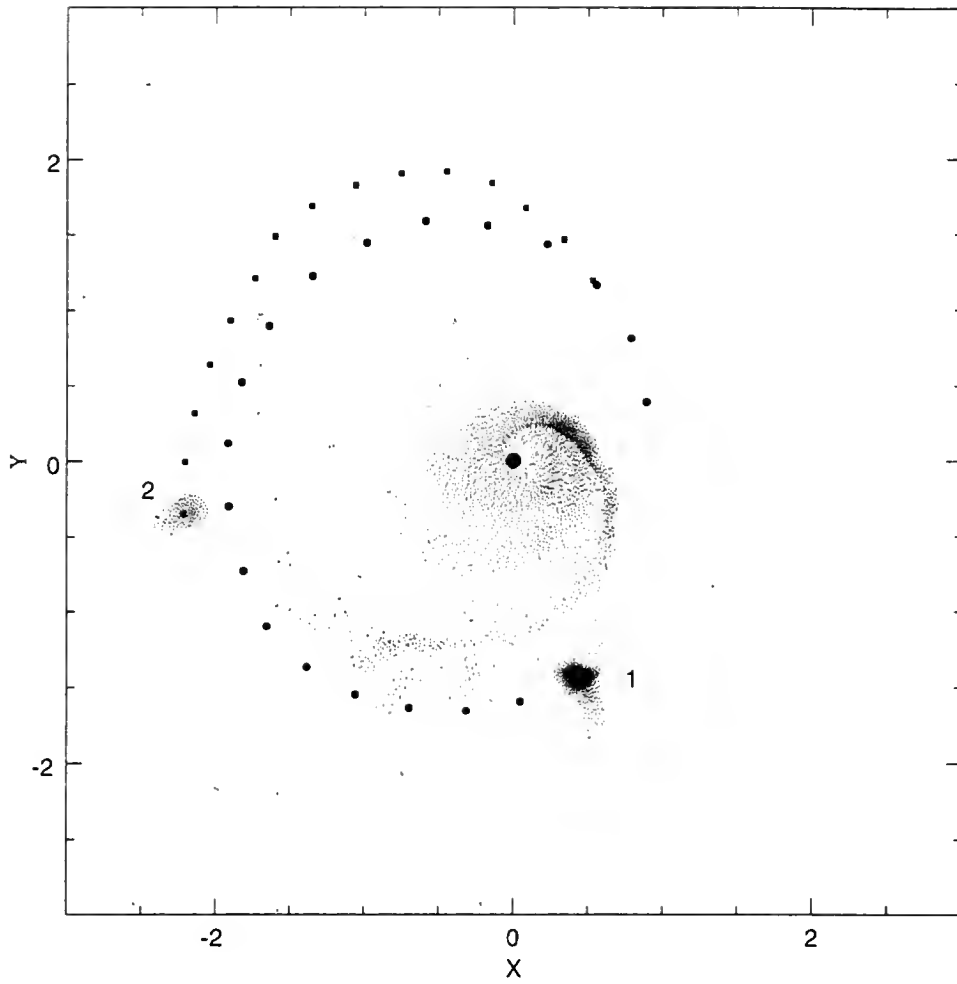


Figure 5-6: Final configuration of model B3, showing the position of the satellites at earlier times.

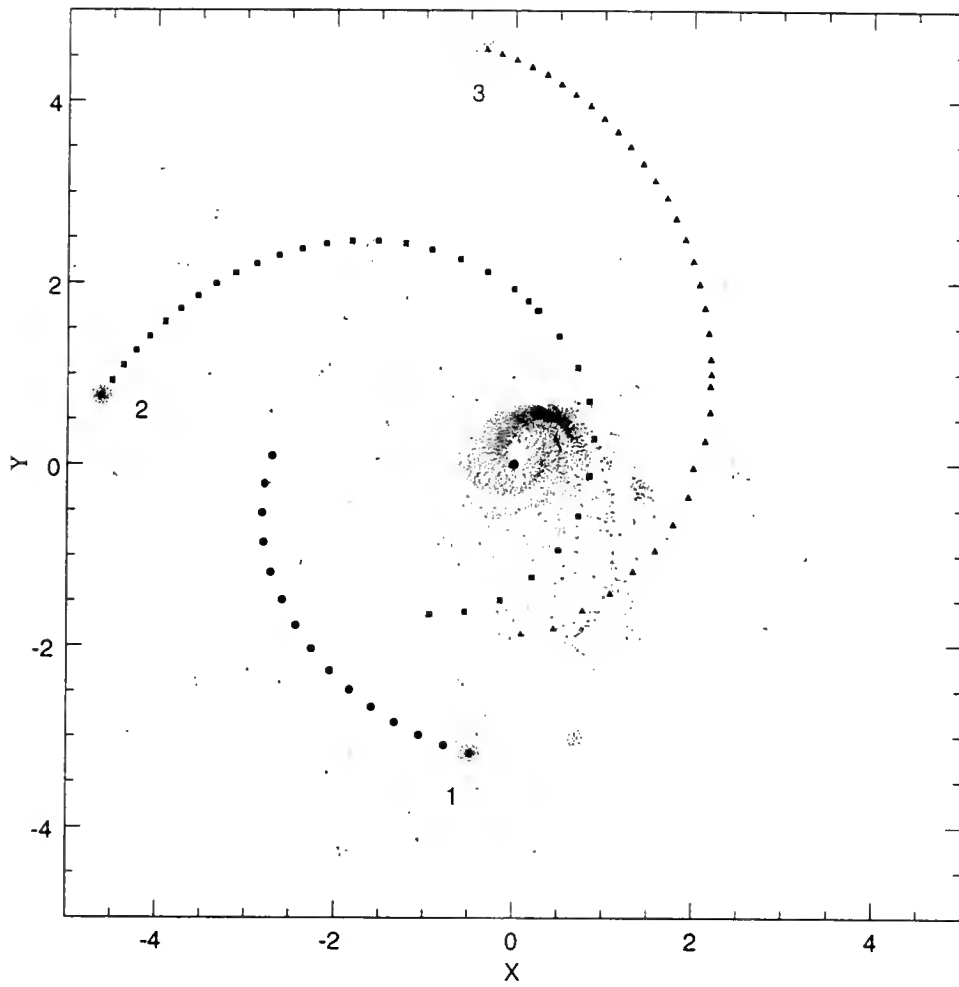


Figure 5-7: Final configuration of model B1, showing the position of the satellites at earlier times.

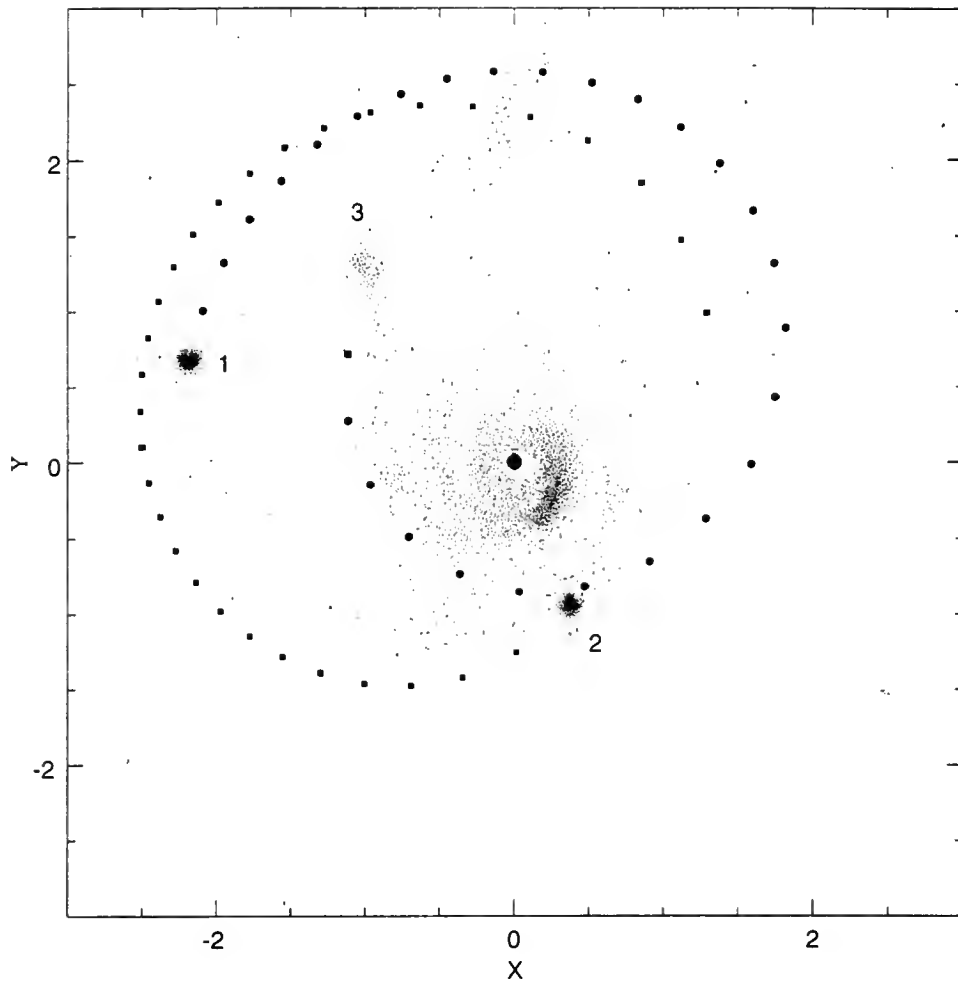


Figure 5-8: Final configuration of model D1, showing the positions of the satellites at earlier times. Satellite 4, which has been reabsorbed by the disk, is not shown.

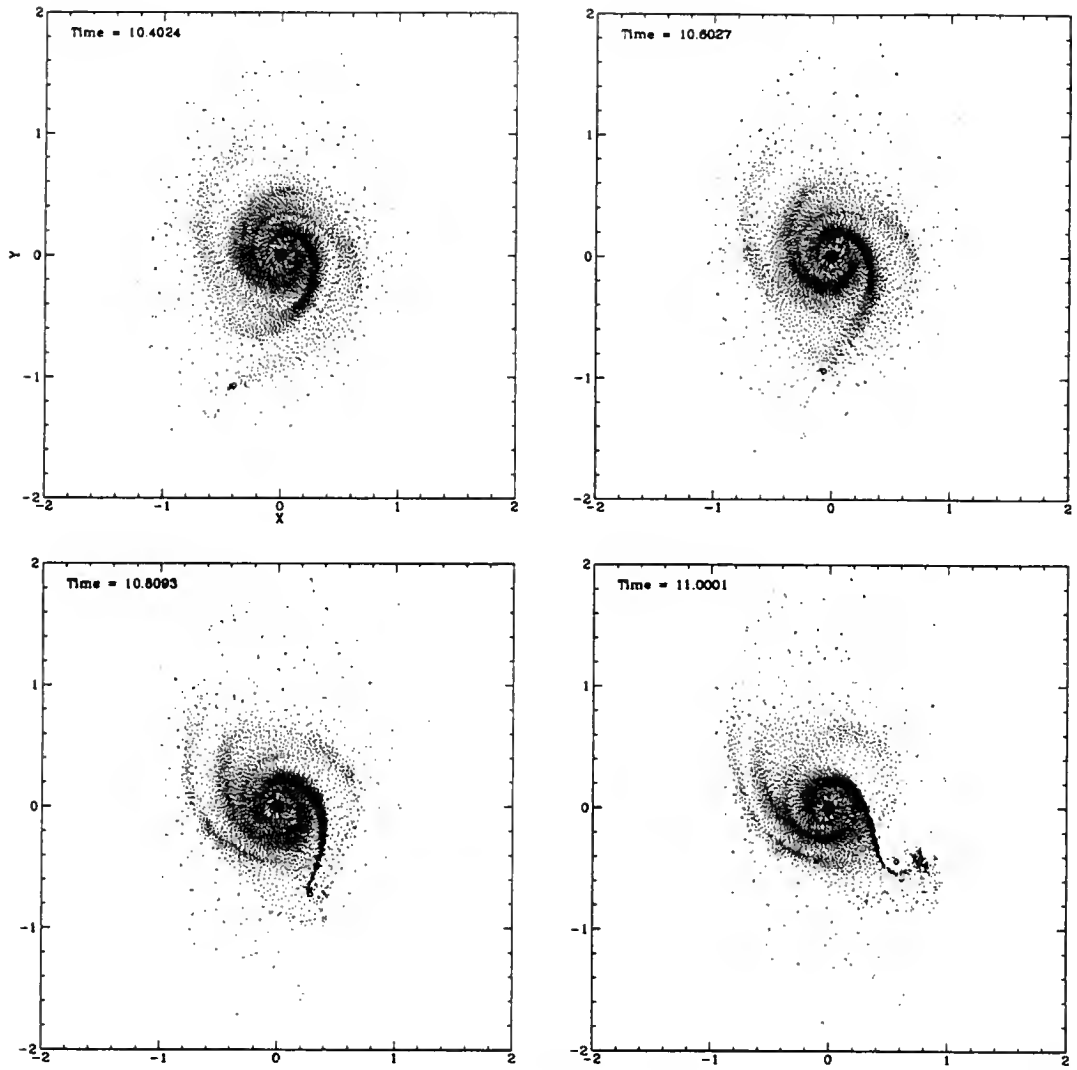


Figure 5-9: Encounter model.

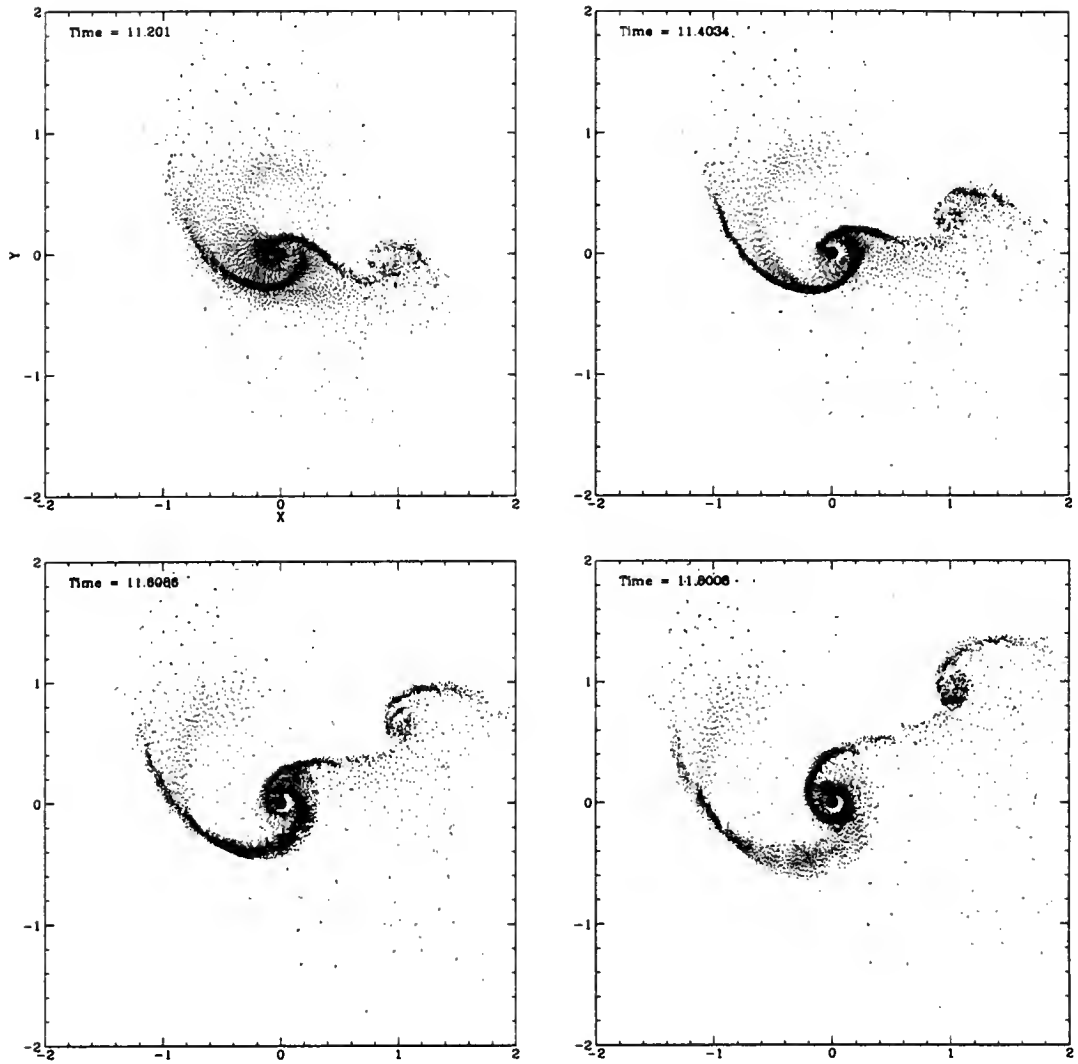


Figure 5-10: Encounter model continued.

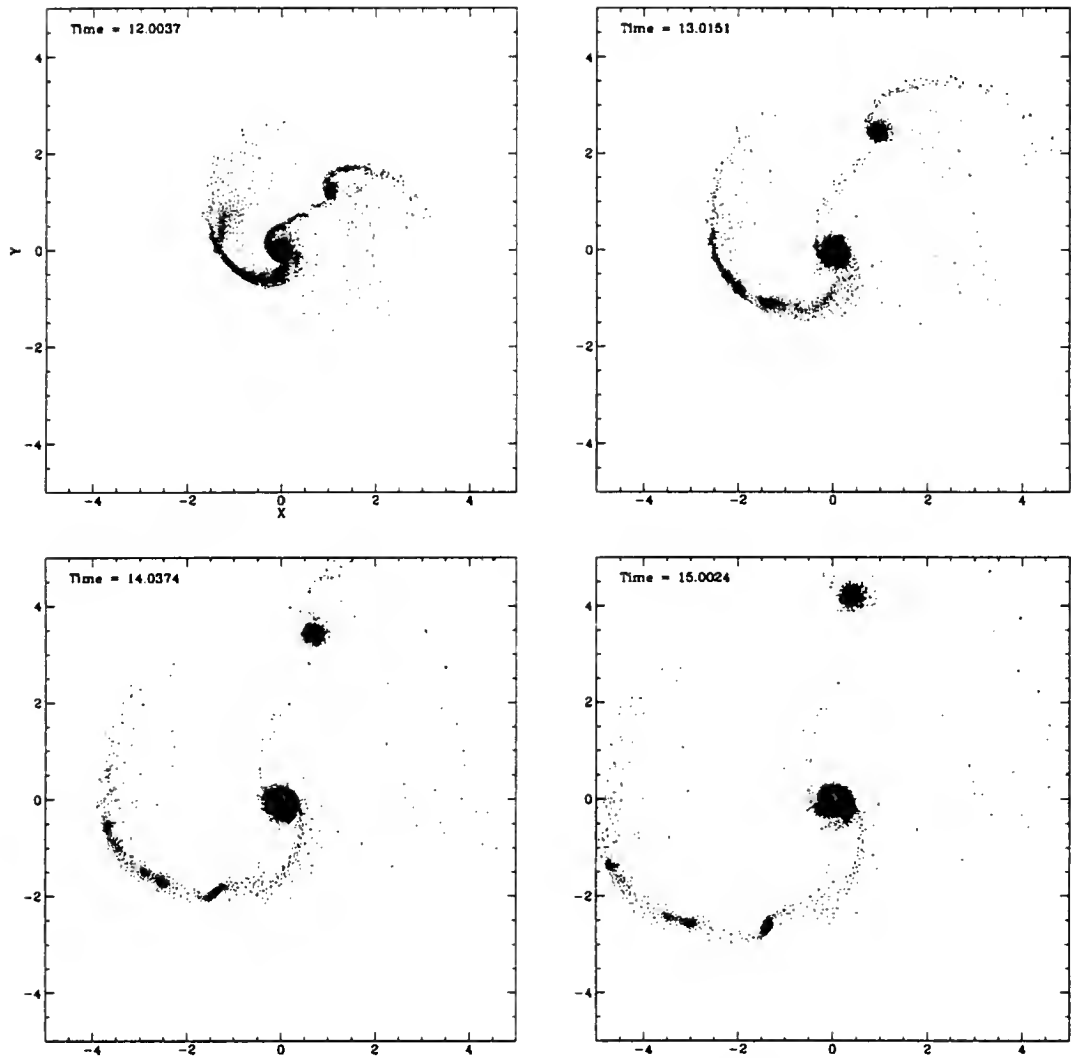


Figure 5-11: Encounter model continued.

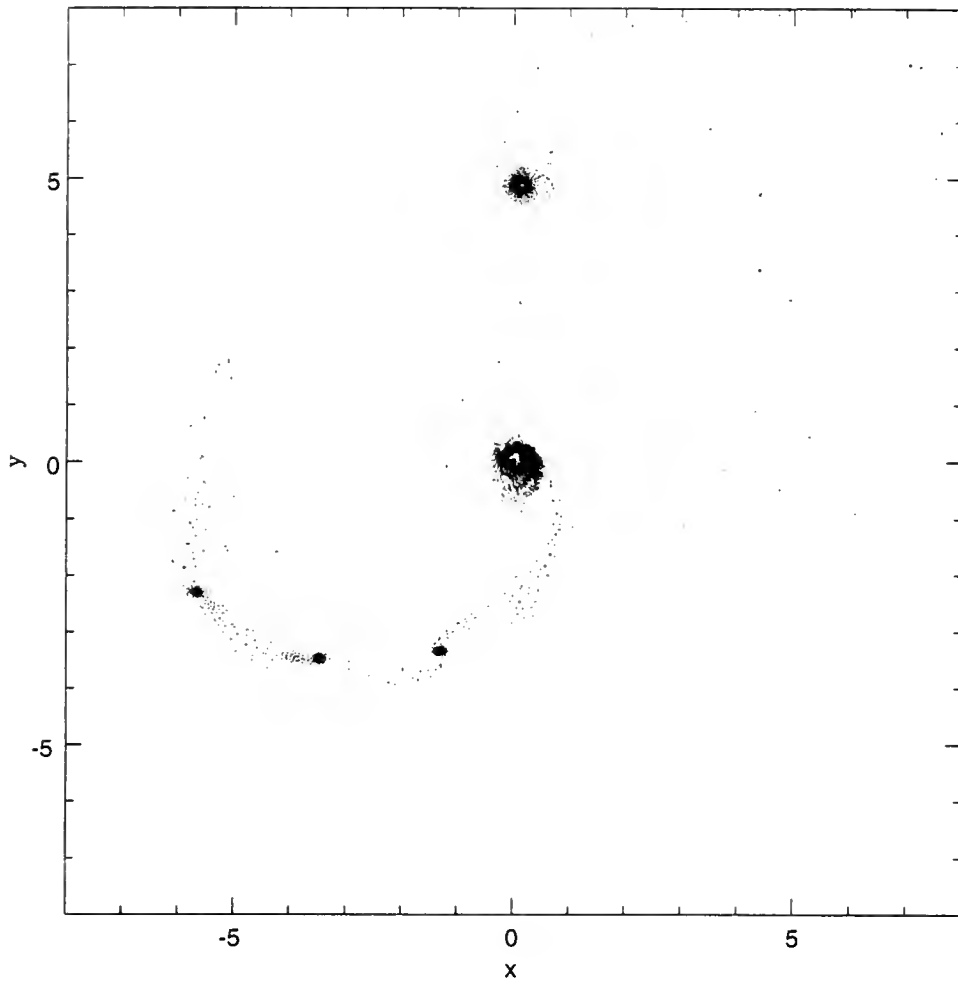


Figure 5-12: Encounter model at Time = 16.0. Note the three satellites that have formed in the tidal tail.

CHAPTER 6 SUMMARY AND CONCLUSIONS

Results

Code Development

The goal of this work is to model accretion disks that are similar to protostellar disks, and to ascertain the effect of shear viscosity on the evolution of these objects. The first step toward this accomplishment was the development of a numerical hydrodynamics computer code which can calculate the gravitational forces, as well as the hydrodynamical forces, that dictate the dynamics of a given system. The numerical hydrodynamics were added to an already existing hierarchical tree N-body code developed by Hernquist (1987), and the method chosen was SPH (smoothed particle hydrodynamics). In implementing this method I follow the standard techniques of SPH that others have already developed and communicated in the literature. In order to efficiently model astrophysical disks, that often display a large dynamic range in time scales, I improved the integrator by incorporating individual time steps. However, certain characteristics of this problem demanded that I modify the method.

The first modification was to impose an inner boundary condition around a central massive particle, that represented the protostar, which would allow the particles that represent the gas to “accrete” onto the central object. At the same time the boundary must give some pressure support to the gas, so that the region around the central object will not be an unrealistic sink for the gas. Other authors have suggested a method for

modeling accretion with SPH (Anzer *et al.*, 1987), but it is unsatisfactory, as the rate of accretion is determined by the parameters of the method. In a real astrophysical disk, accretion is determined by processes taking place throughout the disk, and not by the inner boundary condition; I wish the same to be the case in my models. Therefore I developed an inner boundary condition which circumscribes an inner axisymmetric disk with a density profile that is adjusted so that the boundary condition remains smooth in the density and the velocities. This inner density profile is described by inner SPH particles placed on concentric rings (see Chapter 3). The boundary provides not only the pressure force, but also the viscous force, needed to realistically model accretion. To test that mass accretion is not being determined by the inner boundary condition, I ran models with two different density profiles describing the inner disk (Chapter 4), and found that the mass accretion varied only slightly, indicating a large degree of success. The accuracy of the models is further evidenced by the accretion rates found for the disks which, when transformed to physical units, match those inferred from observations.

Another modification to the SPH method was the development of an artificial viscous term that accurately models shocks, and yet introduces a realistic amount of viscous shear, whose strength can be varied. To achieve this I developed a hybrid version of the standard artificial viscous term already used in standard SPH. Its description, as well as the accretion shock models used to test it, is found in Chapter 2. Also of importance is a clear understanding of the nature of the effective viscous shear produced by this artificial term, which was originally introduced to SPH to model shocks accurately. The shear viscous forces are found to be described by a standard shear viscous coefficient

proportional to the local density, sound speed, and smoothing length:

$$\nu = \alpha_\nu \Sigma c h, \quad (6.1)$$

such that the constant α_ν , corresponding to the α parameter of thin viscous disk theory, is about one tenth the artificial viscosity parameter α . This scaling of the viscous shear coefficient is the same as turbulent viscosity, with h being a scale length, which is the type of viscosity that is suspected to be at work in accretion disks (Shakura & Sunyaev, 1973). To determine this scaling of the shear viscous coefficient, models without self-gravity or pressure support are evolved (see Chapter 2).

In the course of completing this work it became evident that several improvements upon the code may improve its performance. In the future I wish to try a different method of rendering the smoothing kernel symmetric ($W_{ij} = W_{ji}$). In this work the kernel is made symmetric by using the standard cubic-spline kernel with an average smoothing length: $W_{ij} = W(\mathbf{r}_{ij}, h_{ij})$, with $h_{ij} = (h_i + h_j)/2$. Others have reported problems with this form of the kernel (Hernquist & Katz, 1989), and another method which may give better results is to use the form $W_{ij} = \frac{1}{2}(W(\mathbf{r}_{ij}, h_i) + W(\mathbf{r}_{ij}, h_j))$. This form may reduce the amount of noise observed in the density on a small scale, as well as making the code more robust (i.e. adaptable) when strong shocks develop on a short time scale.

Another undesirable feature seen in many of the models, if they are given close inspection, is a tendency for particles to clump in pairs, especially in the outer, diffuse portions of a disk. This is well known in the SPH community as the pair instability, and is the result of the mutual smoothed gravity force between two particles not decreasing in strength faster than the opposing pressure force when the two particles approach on

another (Herant, 1994). If particles can approach close enough to one another, their net attraction will cause them to remain associated. At this point the pair becomes a single particle, but with twice the statistical weight as its single neighbors, when the local estimation of quantities is made. It also has the undesired effect of reducing the number of particles used in the simulation. This instability can be avoided by using a gravitational smoothing parameter ε which is spatially variable. An obvious choice would be to set $\varepsilon_i = h_i$.

Modal Evolution

My original goal, when I began this work, was to study the specific variation of the evolution of the modes, and their frequencies, when the strength of the shear viscosity is varied. However, I have found that a detailed description is sometimes not possible, given the number of modes and frequencies present, as well as their transient character. In addition, the evolution is found to be sensitive both to initial conditions, such as the noise in the initial particle positions, as well as to code parameters, such as the tolerance angle. For these reasons a specific description of the evolution of the modes is found to be not only difficult, but of little practical value. General features of the modal evolution are noted in Chapter 4. One consistent result is the presence of an $m=1$ mode that dominates most of the accretion disks that initially have a central mass to disk mass ratio of one. This result agrees with previous analytical work (Savonije *et al.*, 1992) and has been observed by others. Another important feature to be noted is that the amplitude of the modes are damped when the strength of the shear viscosity is increased.

Accretion

More interesting than the details of the numerical method are the results found from the models themselves. The results in both this subsection and the next share the common factor that they were unanticipated. The mass accretion onto the central object, driven by the local action of shear viscous forces and the global action of the nonaxisymmetric modes, shows three distinct features. First, for most of the disks, the accretion rate after about six to ten dynamical times attains a value constant with time. How an accretion disk regulates itself in order to maintain a constant accretion rate is not clear. The second result is that the accretion rate, after attaining a constant value, sometimes undergoes a brief adjustment period to attain a new constant accretion rate. For the accretion disks that begin with an initial central mass to disk mass ratio of one, the shifts in the accretion rate are correlated to the $m=1$ mode gaining dominance in the disk. The third and most interesting result is that mass accretion, whether measured by the total mass accreted, or the accretion rate, is roughly *inversely* proportional to the strength of the shear viscous forces.

Though counter-intuitive, this last result can be understood by considering the effect of the shear viscosity on the nonaxisymmetric modes. These modes, when present, are more efficient than shear viscosity at transporting angular momentum, and thereby driving accretion. As the shear viscosity increases the amplitude of the nonaxisymmetric modes is damped, resulting in a decline of their efficiency to drive accretion. In the past the action of nonaxisymmetric modes has been modeled in α -disk theory, which assumes axisymmetry and that the disk is thin (see Pringle 1981), as simply an additive effect

(Lin & Pringle, 1987). That is, the presence of nonaxisymmetric modes is modeled by increasing the effective viscosity parameter α_ν . My results suggest that this approach may be too simple, as the two mechanisms for transporting angular momentum are not independent of each other. While it is true that accretion will be greater when nonaxisymmetric modes are present than if they are not, my results suggest that the converse is not true. When shear viscous forces are present, accretion is hindered.

Formation of Satellites

In all disks that formed a dominating $m=1$ mode satellites are also observed to form. Though the dominating $m=1$ mode is not directly responsible for the formation of the satellites, the importance of its eventual development is born out by model D3, which has neither a dominating $m=1$ mode nor satellites. A correlation between the number of satellites formed and the ratio of specific heats, γ , is not clear, whereas, more satellites are formed in models with the least amount of viscous shear. Also, it should be mentioned that the satellites in models B1 and D1, which differ only in their ratio of specific heats, form by different mechanisms. Two of the satellites in model B1 form by a gradual accumulation of particles. This process of formation may be an artifact of the SPH method which, in representing a gas by a finite number of particles, does not resolve diffuse regions well. However, the majority of the satellites form from the outer portions of arms of nonaxisymmetric modes. Therefore, a lower value of γ may favor satellite formation, though this study does not resolve the question. The satellites, once formed, also exhibit vigorous orbital evolution. The orbital evolution of all the satellites are described, and then explained by applying the simple model of two bodies

in circular rotation about one another. Though many of the satellite orbits have significant eccentricity after they form, and sometimes interact with other satellites, the “toy model” explains their overall orbital evolution well.

In modeling astrophysical disks many others have reported the fission of a disk, starting with Lucy (1977) and more recently Bonnell (1994). Others have investigated the fragmentation of collapsing clouds (see Boss, 1992). However, these models begin without a central massive object, and therefore are not accretion disks. The companions formed are massive, and envisioned to be the birth of a companion star. The models presented in this work are the only models which exhibit the formation of small satellites in a disk gravitationally dominated by a central body. Their significance to the theory of planet formation is obvious. Others have pointed out both the necessity and difficulty of Jupiter forming early in the history of the protosolar disk (Wetherill, 1994). Might these satellites, if allowed to complete their collapse, be the seed for large bodies in the outer solar system? If so it would be a mechanism, gravitational in nature, by which planets may be formed, as opposed to the process of accumulation of planetesimals that is currently in favor (Hayashi *et al.*, 1977; Boss, 1989; Wetherill, 1990). However, others have speculated on the presence of vortices in a protostellar disk which may lead to local gravitational instabilities (Hunter & Schweiker, 1981; Hunter & Horak, 1983; Abramowicz *et al.*, 1992).

One difficulty with supposing that the satellites may be protoplanets is that a dissipative mechanism, which would later circularize the orbit, must be found. If such a mechanism does not exist then these models could be used to argue that Jupiter could not

have formed in the way that these satellites form, because it possesses a nearly circular orbit. Also, the masses of the satellites, if $M_T = 1M_{\odot}$, are about an order of magnitude greater than Jupiter's, the smallest satellite being 4.4 Jupiter masses. In addition, the formation of satellites is favored by the two-dimensional nature of these disks; because the mass is restricted to two dimensions, higher densities are achieved than those that would be achieved in three dimensions. If these problems are resolved, then the models suggest the possibility of massive planets at large distances from their "suns", as many of the satellites spiral out to large radii.

Future Work

A natural extension of this work, and the next priority of this research, is to generalize the models to three dimensions. Although the results presented here are intriguing, they must be verified to persist in three-dimensional models, especially since disks with appreciable mass will have a nonnegligible thickness. A few three-dimensional models have been presented, but not enough to provide a thorough study of the effects of viscosity on such disks, much less to ascertain whether the results obtained for the two-dimensional disks will persist.

Another aspect of the modeling that must be modified, to make more realistic models, is the equation of state. By assuming a polytropic equation of state, the need to model thermodynamics is avoided. This approach has been used here, and in both the modeling and analytical work of others, for the sake of necessity and convenience. This approximation enables the theorist to make analytical progress, and makes the interpretation of models much easier. Such is the case in this work as well, though not

for the reason of avoiding yet harder work: scientific progress and understanding takes place in steps. Idealized problems, though academic, must first be explored before more realistic and complex models can be understood. Therefore I think it will be profitable to use the pressure of an ideal gas, in conjunction with evolving the thermal energies of the particles, after three-dimensional models are first evolved with the polytropic equation of state. Adding thermodynamics to SPH is not difficult when the disk is optically thin. However, when the optical depth becomes important at certain wavelengths, then radiative transfer must be included, and cooling will take place at the surface.

In this work magnetic forces were completely neglected, yet they may play an important role in the evolution of a protostellar disk. As was mentioned in the introduction, the presence of a weak magnetic field may be the source of the instability responsible for the anomalous viscosity that protostellar disks must possess to remain short-lived objects. However, others have recently argued that viscosity is limited by causal restraints to much lower values than the lifetimes suggest (Narayan *et al.*, 1994). If this is the case, then SPH will only be sufficient to model such low viscosity disks with higher particle numbers, so that the smoothing length of the particles is significantly smaller than the disk height. In any event, an explicit shear viscous term for SPH is necessary to model a gas with any desired form of viscosity, let alone a viscosity which is independent of the local smoothing length. Flebbe *et al.* (1994) have introduced such a term in an SPH code without a dynamically varying smoothing length. Instead the smoothing length in their version of SPH remains constant.

Another aspect of real protostellar disks that must eventually be taken into account is the environment in which they form. Star formation is observed to occur in crowded environments, and encounters during the lifetime of a disk will not be uncommon. Numerical studies of the effect of encounters with protostellar disks have begun with the work of Heller (1993), who reports on the effect an encounter has upon the tilt of the disk, and Clark and Pringle (1993). In order to draw general conclusions about the effect of encounters, rather than just provide a description of specific cases, the number of encounter models considered must be sufficient to explore the range of possible configurations. This demands a large number of models, as there are three primary controlling parameters: the minimum separation, the angle of approach, and the relative velocity.

Once realistic three-dimensional models are built the next step will be to develop methods to "observe" the models synthetically at various wavelengths, resolutions, and orientations. Such techniques are already being developed (Bouvier, 1994). This will enable specific protostellar disks to be modeled. Such observations of real systems have been made, as mentioned in the first chapter, and more observations of these objects will be made as new instrumentation, such as the Millimeter Array and Hubble Telescope, is employed to search for these objects.

The formation of satellites in the two-dimensional models that I have presented here is intriguing, and it will be interesting to see if they form in corresponding three-dimensional models, or if their formation is aided by cooling when more thermodynamics is included. I also suspect that the formation of the satellites is a chaotic process, in the sense that the

number of satellites and their loci of formation are sensitive to initial conditions, but this should be confirmed with modeling. Also, further analytic work should be developed in the context of satellite formation (eg. a local analysis), such as that begun by Hunter & Horak (1983). Hopefully such analysis will throw light upon the connection between the $m=1$ mode that invariably accompanies satellite formation.

The most interesting, and enlightening, results are those concerning mass accretion. However, these same results also raises new questions. How does an accretion disk regulate mass accretion in order to insure that it remains at a constant rate? That such behavior is preferred is underlined by the observed shifts to new constant accretion rates, which suggest that, when physical conditions in the disk change, which either encourages or hinders accretion, the disk quickly adjusts to accommodate a new constant accretion rate.

APPENDIX A T2DSPH

```

@PROCESS DC (BODYCEL, CONCOM, GAS)
@PROCESS DIRECTIVE ('*VDIR:')
C*****
C
C
C   PROGRAM T2DSPH
C
C               VERSION 2: JUNE 1995
C
C       RONALD DRIMMEL, UNIVERSITY OF FLORIDA
C*****
C
C
C   A CODE TO EVOLVE SELF-GRAVITATING SYSTEMS USING THE HIERARCHICAL
C   TREE METHOD DEVELOPED BY BARNES AND HUT (NATURE 324, 446 [1986])
C   AND IMPLEMENTED IN FORTRAN BY HERNQUIST (AP. J. SUPPL. 64, 715
C   [1987]; COMP. PHYS. COMM. 48, 107 [1988]). THE GASEOUS COMPONENT
C   IS EVOLVED USING SMOOTHED PARTICLE HYDRODYNAMICS (SPH), WHICH WAS
C   ADDED BY RONALD DRIMMEL FOLLOWING, IN PART, HERNQUIST AND KATZ'S
C   OWN UNIFICATION OF TREE-HIERARCHICAL N-BODY CODE WITH SPH (1989).
C   THIS VERSION HAS BEEN OPTIMIZED FOR SUPERCOMPUTERS AND IS FULLY
C   VECTORIZED. THE CODE IS WRITTEN IN STANDARD FORTRAN.
C
C   IN THIS VERSION, VECTORIZATION OF THE TREE WALKS IS ACHIEVED BY
C   SIMULTANEOUSLY PROCESSING ALL CELLS AT THE SAME LEVEL IN THE
C   TREE, AS DISCUSSED BY HERNQUIST (J. COMP. PHYS, SUBMITTED
C   [1988]). THE GRAVITATIONAL FORCE CALCULATION PROCEEDS FOR A
C   SINGLE PARTICLE AT A TIME, IN SERIAL ORDER.
C
C   THE GRAVITATIONAL FIELD IS SMOOTHED ACCORDING TO A CUBIC SPLINE
C   KERNEL. THE KERNEL IS COMPUTED BY LINEAR INTERPOLATION FROM A
C   LOOK-UP TABLE.
C
C   A SELF-STARTING LEAP-FROG INTEGRATOR IS USED, AS DESCRIBED BY
C   HERNQUIST AND KATZ (AP. J. SUPPL., 1989), AND INDIVIDUAL
C   TIME STEPS WAS IMPLEMENTED, AGAIN FOLLOWING HERNQUIST AND
C   KATZ, BY RONALD DRIMMEL. THE TIME BIN OF EACH PARTICLE IS
C   DETERMINED BY SUBROUTINE COURANT, WHICH DEFINES
C   THE TIME STEP BY USING THE COURANT CONDITIONS.
C
C   THE COMPUTATIONAL SYSTEM OF UNITS IS DETERMINED BY THE INPUT
C   DATA, WITH THE ASSUMPTION THAT G=1. PARTICLES ARE NOT
C   REQUIRED TO HAVE IDENTICAL MASSES.

```

```

C
C THIS VERSION USES AN POLYTROPIC EQUATION OF STATE. THE SMOOTHING
C LENGTH FOR EVERY PARTICLE VARIES IN SPACE AND TIME IN ORDER
C TO HAVE A LOCALLY REAJUSTABLE RESOLUTION. PARTICLES FROM 1
C TO NGP ARE GASEOUS WHILE PARTICLES FROM NGP+1----> NBODIES
C ARE COLLISIONLESS. WHEN NGP IS EQUAL TO ZERO WE HAVE A PURE
C NBODY SYSTEM WHILE WHEN NGP=NBODIES WE HAVE A PURE GASEOUS
C SYSTEM. THE INTERPOLATION KERNEL USED FOR THE REPRESENTATION
C OF GASEOUS PROPERTIES IS THE W3 SPLINE FOR 2-D PROBLEMS.
C
C TWO INPUT DATA FILES ARE REQUIRED TO RUN THIS CODE: A PARAMETER
C FILE, WHICH IS READ IN THROUGH THE SUBROUTINE INPARAM, AND A
C BODY DATA FILE, READ BY SUBROUTINE INBODS. BOTH ARE ASCII AND
C THEIR STRUCTURE IS DEFINED IN THE SUBROUTINES WHICH READ THEM.
C THREE OUTPUT FILES ARE CREATED: AN ASCII LOG FILE, AN ASCII
C BODY DATA FILE CONTAINING THE FINAL STATE OF THE SYSTEM, AND
C A BINARY BODY DATA FILE. THE LOG AND BINARY BODY FILES ARE
C UPDATED EVERY NOUTLOG AND NOUTBOD STEPS, RESPECTIVELY.
C THERE IS ALSO A HEADER FILE, CALLED HEADER, WHICH DEFINES
C GLOBAL PARAMETERS AND COMMON BLOCKS.
C
C WARNINGS -- TO AVOID EXCESSIVE OVERHEAD, NOUTLOG SHOULD BE
C LARGER THAN 1, TYPICALLY ^ 10, DEPENDING ON THE
C NUMBER OF STEPS.
C
C WHEN COMPILING ON VAX'S, AVOID USING OPTIMIZATION.
C
C PLEASE REPORT ALL PROBLEMS OR SUGGESTIONS FOR IMPROVEMENTS TO
C THIS CODE TO HERNQUIST@IASSNS.BITNET.
C
C=====
C
C THIS IS THE TOP-LEVEL EVOLUTION PROGRAM TREECODE. ITS TASKS ARE:
C
C 1) TO INITIALIZE FILE STRUCTURES AND GLOBAL VARIABLES;
C 2) TO INPUT PARAMETERS AND THE INITIAL SYSTEM STATE;
C 3) TO ADVANCE THE STATE OF THE SYSTEM FOR A GIVEN NUMBER
C OF TIMESTEPS;
C 4) TO PERFORM A DIAGNOSTIC ANALYSIS OF THE SYSTEM AT
C EACH TIME STEP (ENERGY, ANGULAR MOMENTUM, ETC.);
C 5) TO PERIODICALLY RECORD THE STATE OF THE SYSTEM;
C 6) AND TO TERMINATE THE SIMULATION AND CLOSE DATA FILES.
C
C=====
C

```

```

C
C   BASIC GLOBAL VARIABLES/PARAMETERS:
C
C       ACC           : ACCELERATION COMPONENTS OF A BODY.
C       ACCLIST       : LIST OF ACCRETED PARTICLES
C       ACSMOOT       : TABLE OF SMOOTHED GRAVITATIONAL ACCELERATION.
C       ADVLIST       : LIST OF PART. WHOSE VELOCITIES ARE ADVANCED.
C       ALFA          : VISCOSITY PARAMETER.
C       AMVEC         : ANGULAR MOMENTUM VECTOR.
C       ANW           : WIDTH OF ANNULI USED FOR AM BOOKKEEPING.
C       ARAD          : ACCRETION RADIUS
C       BNOW          : CURRENT TIME BIN
C       BHTA          : VISCOSITY PARAMETER
C       C             : SOUND SPEED AT EACH GAS PARTICLE
C       CELLSIZ       : LINEAR SIZES OF CELLS.
C       CMPOS         : POSITION OF THE CENTER OF MASS.
C       CMVEL         : VELOCITY OF THE CENTER OF MASS.
C       CN            : COURANT NUMBER (TIME STEP PARAMETER).
C       CPUTIME       : CPU TIME (SECS) USED DURING THE SIMULATION.
C       CPUT0         : CUMULATIVE CPU TIME AT START OF RUN.
C       CPUT1         : CUMULATIVE CPU TIME AT END OF RUN.
C       DELV          : DEL DOT V AT EACH GAS PARTICLE
C       DRHDT         : PROJECTED RATE OF DENSITY CHANGE.
C       DT            : THE INDIVIDUAL TIME STEP.
C       DTIME         : THE CURRENT TIMESTEP.
C       DTIME2        : DTIME/2.
C       DTS           : SYSTEM TIME STEP.
C       DVEL          : INITIAL VELOCITY.
C       EPS           : GRAVITATIONAL SMOOTHING PARAMETER.
C       EKTOT         : TOTAL SYSTEM KINETIC ENERGY.
C       EPTOT         : TOTAL SYSTEM GRAVITATIONAL POTENTIAL ENERGY.
C       ETOT          : TOTAL ENERGY OF THE SYSTEM.
C       F             : FACTOR.
C       GAMMA         : EXPONENT IN EQUATION OF STATE
C       H             : SMOOTHING LENGTH OF GASEOUS PARTICLES
C       HEADLIN       : IDENTIFICATION STRING FOR THE RUN.
C       HMAX          : MAXIMUM SMOOTHING LENGTH.
C       INCELLS       : NUMBER OF CELLS CURRENTLY IN USE.
C       ISS           : ISOTHERMAL SOUND SPEED (GAS CONSTANT)
C       LIST          : LIST OF NEAREST NEIGHBORS
C       LT(I)         : ANGULAR MOMENTUM TRANSPORTED ACCROSS RADIUS I.
C       MASS          : MASSES OF BODIES AND CELLS.
C       MDOT          : MASS ACCRETION RATE FROM RADIAL VELOCITIES.
C       MINDX         : ANNULUS INDEX OF PARTICLES.
C       MINUST        : THE CONSTANT -2.
C       MTOT          : TOTAL MASS OF THE SYSTEM.
C       NACC          : TOTAL NUMBER OF ACCRETED PARTICLES
C       NADVLIS       : NUMBER OF PARTICLES IN ADVLIST.

```

C	NAGP	: NUMBER OF ACCRETED GAS PARTICLES
C	NBODIES	: TOTAL NUMBER OF INITIAL BODIES
C	NBODLIS	: TOTAL NUMBER OF ACTIVE BODIES
C	NBODSMA	: MAXIMUM NUMBER OF BODIES.
C	NBSLIST	: LIST OF PARTICLES IN THE SAPLING.
C	NCELLS	: MAXIMUM NUMBER OF CELLS.
C	NCP	: NUMBER OF CENTRAL PARTICLES
C	NDIM	: NUMBER OF SPATIAL DIMENSIONS.
C	NGAS	: NUMBER OF GAS PARTICLES IN CURRENT TIME BIN.
C	NGP	: INITIAL NUMBER OF GASEOUS PARTICLES.
C	NGPMAX	: MAXIMUM NUMBER OF GAS PARTICLES.
C	NGPTS	: NUMBER OF GRID POINTS.
C	NINTERP	: NUMBER OF VALUES IN LOOK-UP TABLES.
C	NMAX	: MAXIMUM CURRENT TIME BIN.
C	NMP	: NUMBER OF MASSIVE (ACCRETING) PARTICLES
C	NNB	: NUMBER OF NEAREST NEIGHBORS
C	NNBS	: NUMBER OF NEIGHBORS WITHIN H/SOF.
C	NNBMAX	: OPTIMUM NUMBER OF NEIGHBORS.
C	NNBSLIS	: NUMBER OF PARTICLES IN NBSLIST.
C	NOUTBOD	: FREQUENCY OF SYSTEM RECORD OUTPUTS.
C	NOUTLOG	: FREQUENCY OF OUTPUTS TO LOG FILE.
C	NSTEPS	: NUMBER OF SYSTEM TIMESTEPS IN THE SIMULATION.
C	NSUBCEL	: NUMBER OF SUBCELLS PER CELL.
C	NTAVG	: AVERAGE LENGTH OF INTERACTION LISTS.
C	NTMAX	: LARGEST INTERACTION LIST IN CURRENT TIME STEP.
C	NTMIN	: SHORTEST INTERACTION LIST IN CURRENT TIME STEP.
C	NTTOT	: SUM OF INTERACTION LISTS IN CURRENT TIME STEP.
C	ONE	: THE CONSTANT 1.
C	PHI	: GRAVITATIONAL POTENTIAL.
C	PHSMOOT	: TABLE OF LOOK-UP VALUES FOR GRAV. POTENTIAL.
C	PI	: CONSTANT PI.
C	POS	: COORDINATES OF BODIES, C.M. COORDS. OF CELLS.
C	QTOT	: TOTAL ENERGY LOST.
C	QUAD	: QUADRUPOLE MOMENTS OF CELLS.
C	RBMIN	: COORDS. OF LOWER-LEFT CORNER OF SYSTEM BOX.
C	RH	: GAS DENSITY
C	RHS	: SMOOTHED DENSITY.
C	RMAX(N)	: MAXIMUM RADIAL EXTENT OF PARTICLES TIME BIN N.
C	RMIN(N)	: MINIMUM RADIAL EXTENT OF PARTICLES TIME BIN N.
C	ROOT	: POINTER TO THE TOP OF THE TREE.
C	RSIZE	: LENGTH OF THE SYSTEM BOX.
C	SLIST	: LIST OF NEIGHBORS WITHIN H/SOF.
C	SOF	: DENSITY SOFTENING PARAMETER.
C	SUBP	: POINTER TO DESCENDENTS OF A CELL.
C	TAAM	: TOTAL ACRETED ANGULAR MOMENTUM
C	TAM	: TOTAL ACRETED MASS
C	TAEK	: TOTAL ACRETED KINETIC ENERGY
C	TBIN	: LIST OF TIME BIN OF EACH PARTICLE.

C TD : TIME INTERVAL BETWEEN OUTPUTS
 C TDUMP : TIME OF NEXT OUTPUT
 C TELAPS : TIME ELAPSED DURING CURRENT SYSTEM TIME STEP.
 C TINY : A SMALL NUMBER USED TO PREVENT DIVERGENCES.
 C TNOW : CURRENT SYSTEM TIME.
 C TOL : ACCURACY PARAMETER.
 C TOL2INV : $1. / (TOL * TOL)$.
 C TPOS : CURRENT POSITION TIME.
 C TSTEP : TOTAL NUMBER OF TIME STEPS.
 C TWO : THE CONSTANT 2.
 C U : PARTICLE THERMAL ENERGY.
 C USEQUAD : OPTION TO USE (.TRUE.) QUADRUPOLE TERMS.
 C UDOT : RATE OF THERMAL ENRGY CHANGE.
 C UTOT : TOTAL THERMAL ENERGY.
 C VEL : VELOCITY COMPONENTS OF A BODY.
 C VET : ESTIMATED CURRENT VELOCITY OF GAS PARTICLE.
 C VTYP : VISCOSITY TYPE
 C ZERO : THE CONSTANT 0.
 C ZETA : SMOOTHING LENGTH ADJUSTMENT PARAMETER.

DEFINITIONS SPECIFIC TO INPUT/OUTPUT.

C UTERM, UPARS, ULOG, UBODSIN, : LOGICAL I/O UNIT NUMBERS.
 C UBODSOU,UBODSAS
 C PARSFIL, LOGFILE, IBODFIL, : CHARACTER NAMES OF FILES.
 C OBODFIL,OASCFIL

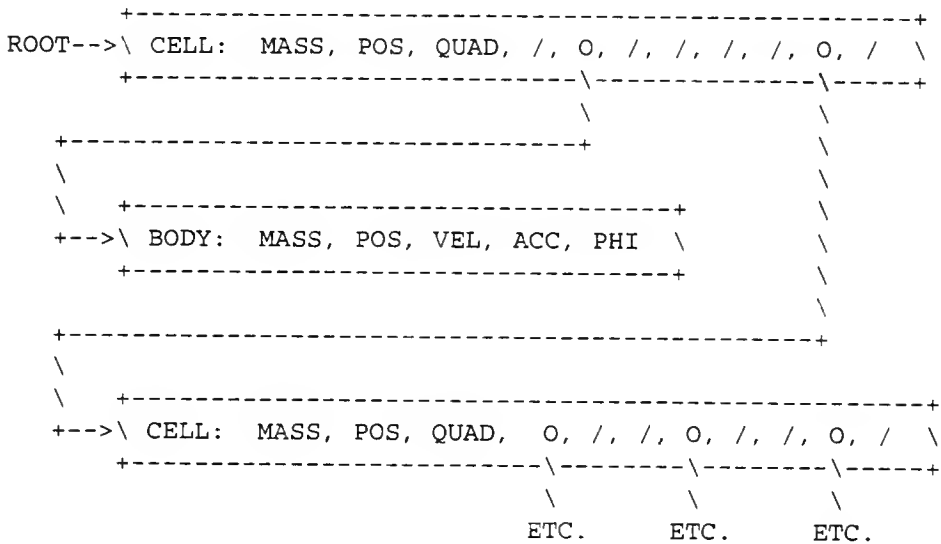
DEFINITIONS SPECIFIC TO VECTORIZED TREE CONSTRUCTION, VECTORIZED TREE WALK, AND VECTORIZED TREE SEARCH FOR NEAREST NEIGHBORS. NOTE THAT BOTTOM IS EQUIVALENCED TO POS, WHICH IS DEFINED ABOVE.

C ASUBP : SUBPOINTER FOR ACTIVE BODIES OR CELLS.
 C ACTLIST : LIST OF ACTIVE BODIES (I.E. NOT YET LEAVES).
 C BODLIST : LIST OF ACTIVE BODIES (I.E. NOT ACCRETED).
 C CELLIST : LIST OF CELLS.
 C ISUBSET : INDICES OF SUBSETS OF ACTIVE BODIES OR CELLS.
 C NWORKVE : LENGTH OF TEMPORARY WORK ARRAY WORKVEC. IT
 C SHOULD BE SET TO 9*MAX LENGTH OF GRAV
 C INTERACTION LIST.
 C PARENT : PARENTS OF ACTIVE BODIES OR CELLS.
 C SUBINDE : SUBINDICES FOR ACTIVE BODIES OR CELLS.
 C SUBPVEC : VECTOR EQUIVALENCED TO SUBP.
 C TEMPLIS : TEMPORARY VECTOR TO SWAP ARRAYS.

WORKVEC : TEMPORARY WORK ARRAY.

DATA STRUCTURE USED TO COMPUTE GRAVITATIONAL FIELD:

THE BODY/CELL TREE STRUCTURE IS ASSUMED TO BE OF THE FORM DISCUSSED BY BARNES AND HUT. SCHEMATICALLY, FOR THREE DIMENSIONS (I.E. EIGHT SUBCELLS PER CELL):



THE BODY/CELL INFORMATION IS STORED IN ARRAYS WHICH INCORPORATE BOTH BODIES AND CELLS. FOR PHYSICAL QUANTITIES RELEVANT TO BOTH BODIES AND CELLS, SUCH AS MASS AND POSITION, THE ARRAY INDICES RANGE FROM 1 --> NBODSMA + NCELLS. FOR THOSE QUANTITIES DEFINED ONLY FOR BODIES, SUCH AS VELOCITY, THE ARRAY INDICES RANGE FROM 1 --> NBODSMA. FOR INFORMATION RELEVANT TO CELLS ALONE, SUCH AS POINTER TO DESCENDANTS, THE ARRAY INDICES RANGE FROM NBODSMA + 1 --> NBODSMA + NCELLS. WITH THIS CONVENTION, POINTER CAN REFER TO EITHER BODIES OR CELLS WITHOUT CONFLICT.

THE IDENTIFICATION OF A GIVEN UNIT AS A BODY OR A CELL IS DETERMINED BY THE POINTER TO THE BODY/CELL. FOR A BODY, P IS LESS THAN OR EQUAL TO NBODSMA, WHILE FOR A CELL, P > NBODSMA.

```

C
C=====
/*INCLUDE NEWHEAD FORTRAN A

C  DECLARATION OF LOCAL VARIABLES.
C  -----

      INTEGER N,NOLD

C=====

C  INITIALIZE STATE OF THE SYSTEM.
C  -----
      N=0

      CALL INITSYS(N)
C  -----
      NOLD=N

C  ADVANCE SYSTEM STATE FOR A GIVEN NUMBER OF STEPS.
C  -----
      TSTEP=1
      DOWHILE(N.LE.NSTEPS)

          CALL STEPSYS(N)
C  -----
          IF(N.NE.NOLD) THEN

              IF(MOD(N,3).EQ.0) CALL GRIDEN
C  -----
              CALL OUTSTAT(NOLD)
C  -----
              NOLD=N
              ENDIF
              TSTEP=TSTEP+1
          ENDDO

C  TERMINATE THE SIMULATION.
C  -----
      CALL ENDRUN
C  -----

      STOP
      END
@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C

```

SUBROUTINE ACCGRAV(OPTION)

```

C
C
C*****
C
C
C      SUBROUTINE TO COMPUTE THE GRAVITATIONAL ACCELERATION FOR ALL OF
C      THE BODIES.  VECTORIZATION IS ACHIEVED BY PROCESSING ALL OF THE
C      CELLS AT A GIVEN LEVEL IN THE TREE SIMULTANEOUSLY.  THE LOCAL
C      VARIABLE OPTION INDICATES WHETHER THE CODE IS TO COMPUTE THE
C      POTENTIAL AND/OR ACCELERATION.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C  DECLARATION OF LOCAL VARIABLES.
C  -----

      CHARACTER*4 OPTION
      INTEGER I,NTERMS,P,NUM
      DOUBLE PRECISION R,A

C=====

C
C  IMPOSED KEPLERIAN POTENTIAL.
C-----
C      NUM=0
C      IF(ADVLIST(1).EQ.NMP) NUM=1
C      IF(OPTION.NE.'POT ') THEN
C          DO I=1+NUM,NADVLIST
C              P=ADVLIST(I)
C              R=DSQRT(POS(P,1)**2+POS(P,2)**2)
C              A=-MASS(1)/R**3
C              ACC(P,1)=A*POS(P,1)
C              ACC(P,2)=A*POS(P,2)
C          ENDDO
C      ENDIF
C      IF(OPTION.NE.'ACC ') THEN
C          DO I=1+NUM,NADVLIST
C              P=ADVLIST(I)
C              PHI(P)=-2.D0*MASS(1)/DSQRT(POS(P,1)**2+POS(P,2)**2)
C          ENDDO
C      ENDIF
C      GOTO 99

```

```

C   INITIALIZE THE INTERACTION LIST DIAGNOSTICS.
C   -----
C       NTTOT=0
C       NTMIN=NBODSMA
C       NTMAX=0

C       IF(OPTION.NE.'POT ') THEN
C           CALL TOLTEST
C           CALL TERROR(' FINISHED WITH TOLERANCE TEST. ')
C       MAIN LOOP OVER ALL BODIES.
C       -----

C           DO 100 I=1,NADVLIS
C               P=ADVLIST(I)

C   ESTABLISH INTERACTION LISTS.
C   -----
C       CALL TREEWAL(P,NTERMS)
C       -----

C   COMPUTE POTENTIAL AND/OR ACCELERATION.
C   -----
C       CALL GRAVSUM(P,NTERMS,OPTION)
C       -----

C   UPDATE DIAGNOSTICS, SUBTRACTING SELF-INTERACTION TERM.
C   -----
C       NTERMS=NTERMS-1
C       NTTOT=NTTOT+NTERMS
C       NTMIN=MIN (NTMIN,NTERMS)
C       NTMAX=MAX (NTMAX,NTERMS)

100      CONTINUE
        ELSE

            DO I=1,NBODLIS

C   ESTABLISH INTERACTION LISTS.
C   -----
C       CALL TREEWAL(BODLIST(I),NTERMS)
C       -----

C   COMPUTE POTENTIAL AND/OR ACCELERATION.
C   -----
C       CALL GRAVSUM(BODLIST(I),NTERMS,OPTION)
C       -----

C   UPDATE DIAGNOSTICS, SUBTRACTING SELF-INTERACTION TERM.

```

```

C      NTERMS=NTERMS-1
C      NTTOT=NTTOT+NTERMS
C      NTMIN=MIN (NTMIN, NTERMS)
C      NTMAX=MAX (NTMAX, NTERMS)
C
C      ENDDO
C
C      ENDIF
C
C      COMPUTE AVERAGE NUMBER OF FORCE TERMS PER BODY.
C      -----
C      NTAVG=NTTOT/NBODLIS
C
C
C      99      RETURN
C      END
C      @PROCESS DC (BODYCEL, CONCOM, GAS)
C      *****
C
C      SUBROUTINE ACCRET(N)
C
C      *****
C
C      SUBROUTINE TO SEARCH FOR PARTICLES THAT HAVE ACCRETED ONTO
C      THE MASSIVE ACCRETING BODIES; TO UPDATE RUNNING TOTALS OF
C      TOTAL ACCRETED MASS, KINETIC ENERGY, AND ANGULAR MOMENTUM;
C      AND TO REMOVE ACCRETED PARTICLES FROM BODLIST.
C      R. DRIMMEL 3/92
C
C      =====
C
C      /*INCLUDE NEWHEAD FORTRAN A
C
C      DECLARATION OF LOCAL VARIABLES
C
C      -----
C      DOUBLE PRECISION AMASS, AAMV(3), AEK, R, COSA, SINA, VR, VT, VX, VY,
C      &      A, THETA, X, Y, B, MR, AMJ, MGP, DTH, X0, Y0, VX0, VY0, RFAC, MG,
C      &      NBDR (NSTMAX), D, ROLD, RL, DR, RM, DM, DMP, ODEN, HP, ACCE,
C      &      FN, DF, DEN, AMDA, V0, SX, SX2, SDN, FAC, VT2, V02, RRNG(20),
C      &      SXD, SVR, SVX, MT, LND0, RSINV, DEN0, THOLD, RAD, ORSI, AAM

```

```

      INTEGER N,J,I,P,NVAL,NS,NPA,INDX,FLAG,NAP,NUM,PP,FLG2,THOS,
&      NINDX,NON,K,NRNG(20),NRG
      DIMENSION INDX(2000),NAP(2000),THOS(2000)
      CHARACTER*4 OPTION
      DATA FAC,THOLD,ODEN,ORSI,RRNG,NRNG,AAM/1.4D0,44*0.D0/,NRG/0/
      SAVE FAC,THOLD,DEN0,ORSI,RRNG,NRNG,AAM,NRG

C THIS STATEMENT IS FOR WHEN THERE IS NO ACCRETION:
C-----
C      GOTO 99
      IF(NMP.EQ.0) GOTO 99

      CALL MAKETRE
C
C      FIND ACCRETED PARTICLES AND MODIFY RUNNING TOTALS
C-----
      NPA=0
      AMASS=0.D0
      AMJ=0.D0
      AEK=0.D0
      ACCE=0.D0
      AAMV(3)=0.D0
      AMDA=0.D0
      MDOT=0.D0
      X0=POS(1,1)
      Y0=POS(1,2)
      VX0=VET(1,1)
      VY0=VET(1,2)
C IF THERE IS AN ENCOUNTERING PARTICLE:
C-----
      IF(NBODIES-NGP.EQ.1) THEN
          CALL WALKIT(NBODIES,H(NBODIES),NVAL)
          DO I=1,NVAL
              NPA=NPA+1
              P=TEMPLIS(I)
              NAP(NPA)=P
              TBIN(P)=-1
              MASS(NBODIES)=MASS(NBODIES)+MASS(P)
          ENDDO
      ENDIF
      FLAG=0
55      NON=0
      CALL WALKIT(1,FAC*ARAD,NVAL)
C      -----
      FLAG=FLAG+1
      DO I=1,NVAL
          IF(WORKVEC(I).GT.ARAD) THEN
              NON=NON+1
          
```

```

      THOS(NON)=TEMPLIS(I)
    ENDIF
  ENDDO
  IF(FLAG.GT.12.OR.FAC.LE.1.D0) THEN
    WRITE(ULOG,*) N,NON,NVAL,FLAG,FAC
    CALL ERROR(' BAD FACTOR. ')
  ENDIF
  IF(NON.LT.50) THEN
    FAC=FAC+0.1D0
    GOTO 55
  ENDIF
  IF(NON.GT.900) THEN
    FAC=FAC-0.1D0
    GOTO 55
  ENDIF
  IF(NON.NE.NVAL) THEN
    NUM=0
    DO I=1,NVAL
      IF(WORKVEC(I).LE.ARAD) THEN
        NUM=NUM+1
        INDX(NUM)=TEMPLIS(I)
      ENDIF
    ENDDO
    DO I=1,NUM
      P=INDX(I)
      NPA=NPA+1
      VX=VET(P,1)
      VY=VET(P,2)
      X=POS(P,1)
      Y=POS(P,2)
      DX=X-CMPOS(1)
      DY=Y-CMPOS(2)
      DVX=VX-CMVEL(1)
      DVY=VY-CMVEL(2)
      TAAM(3)=TAAM(3)+MASS(P)*(X*VY-Y*VX)
      AAMV(3)=AAMV(3)+MASS(P)*(DX*DVY-DY*DVX)
      AAM=AAM+MASS(P)*((X-X0)*(VY-VY0)-(Y-Y0)*(VX-VX0))
      AEK=AEK+0.5D0*MASS(P)*(VX*VX+VY*VY)
      AMJ=AMJ+MASS(P)
      IF(N.EQ.0) THEN
        ACCLIST(NPA)=P
      ELSE
        R=DSQRT((X-X0)**2+(Y-Y0)**2)
        ACCE=ACCE+(U(P)+0.5D0*(VX*VX+VY*VY)
&        -2.D0*MASS(1)/R)*MASS(P)
        MASS(P)=0.D0
        H(P)=0.D0
        U(P)=0.D0

```

```

        ENDIF
        TBIN(P)=-1
        NAP(NPA)=P
    ENDDO
    QTOT=QTOT+ACCE
    AMASS=AMASS+AMJ
    IF (TNOW.EQ.0.D0.OR.N.NE.0) MASS(1)=MASS(1)+AMJ
ENDIF
IF (N.EQ.0) THEN
    ACCE=0.D0
    AMASS=0.D0
    MDOT=0.D0
    AEK=0.D0
    AAMV(3)=0.D0
    TAAM(3)=0.D0
    AAM=U(1)
    NCP=NPA
    NACC=NPA
    IF (NACC.GT.1000) CALL TERROR(' OVERFLOW IN ACCLIST. ')
ENDIF

IF (NPA.NE.0) THEN

C
C   MODIFY BODLIST  (THIS IS NOT THE MOST EFFICIENT METHOD)
C-----
    NAP(NPA+1)=NBODIES+1
    NS=0
    CALL NPKSRT((NPA+1),NAP,INDX)
    NBODLIS=NBODLIS-NPA
    DO I=NMP+1,NBODLIS
        DOWHILE (BODLIST(I+NS).EQ.NAP(INDX(NS+1)).AND.NS.LE.NPA)
            NS=NS+1
        ENDDO
        BODLIST(I)=BODLIST(I+NS)
    ENDDO
    NAGP=NAGP+NPA

ENDIF

C
C   THESE NEXT STATEMENTS ARE FOR THE CASE OF NO INNER DISK
C-----
C       TAM=TAM+AMASS
C       NACC=0
C       NCP=0
C       WRITE(18,*) N,NPA,MASS(1),TNOW
C       GOTO 25

```

```

SX=0.D0
SX2=0.D0
SDN=0.D0
SXD=0.D0
SVR=0.D0
SVX=0.D0
DO I=1,NON
  P=THOS(I)
  X=POS(P,1)-X0
  Y=POS(P,2)-Y0
  R=DSQRT(X*X+Y*Y)
  VR=(Y*(VET(P,2)-VY0)+X*(VET(P,1)-VX0))/R
  SX=SX+R
  SX2=SX2+R*R
  SDN=SDN+DLOG(RH(P))
  SXD=SXD+DLOG(RH(P))*R
  SVR=SVR+VR
  SVX=SVX+VR*R
ENDDO
D=1.D0/(DFLOAT(NON)*SX2-SX*SX)
RSINV=(DFLOAT(NON)*SXD-SDN*SX)*D
LND0=(SDN*SX2-SXD*SX)*D
DEN0=DEXP(LND0)
DEN=DEN0*DEXP(ARAD*RSINV)
MG=2.D0*PI*(DEN0-DEN*(-ARAD*RSINV+1.D0))/RSINV**2
IF(RSINV.GT.0) THEN
  DEN0=DEN
  RSINV=0.D0
  MG=PI*DEN0*ARAD**2
ENDIF
B=(SVR*SX2-SVX*SX)*D
A=(DFLOAT(NON)*SVX-SVR*SX)*D
VR=A*ARAD+B
C NOTE: THIS ASSUMES A SCALE LENGTH OF 0.25 FOR AN OUTER EXP. DISK.
X=ARAD*0.5D0/0.25D0
RFAC=-((0.5772157D0+DLOG(X*0.5D0))*(1.D0+0.75D0*X**2)-1.D0
A=-4.D0*PI*DEN*DEXP(ARAD/0.25D0)*X**2*RFAC*0.25D0/ARAD
V02=MASS(1)/ARAD+A*ARAD
VT2=V02+GAMMA*ISS*ARAD*RSINV*DEN**GMONE
RFAC=DSQRT(VT2/V02)
MDOT=-2.D0*PI*ARAD*VR*DEN
IF(MDOT.LT.0.D0) MDOT=0.0
MASS1=MASS(1)
WRITE(18,100) MASS1,DEN0,MDOT,RSINV,AAM,NPA,TNOW
100 FORMAT(1X,5(1PE13.6,1X),I3,1X,1PE13.6)
MGP=MG/DFLOAT(NACC)
TAM=TAM+AMASS
IF(ABS(DEN0-ODEN).LT.1.D-04.AND.ABS(RSINV-ORSI).LT.1.D-04) GOTO 15

```

```

ODEN=DEN0
ORSI=RSINV

      IF(RSINV.EQ.0.D0) THEN
C IF RSINV IS EQUAL TO ZERO THEN BUILD A UNIFORM DISK:
C   FIND FIRST RADIUS.
C-----
      R=ARAD-DSQRT(MGP/DEN0)*0.5D0
      DR=(ARAD-R)*2.D0
      RM=R-DR*0.5D0
      MR=PI*DEN0*RM**2
      NUM=IDNINT((MG-MR)/MGP)
      MR=MG-NUM*MGP
      RM=DSQRT(MR/(PI*DEN0))
      DR=ARAD-RM
      R=RM+DR*0.5D0
      PP=NUM
      RRNG(1)=R
      NRNG(1)=NUM
      NRG=1
C
C   FIND RADII OF RINGS, OUTSIDE IN.
C-----
      FLG2=0
      DOWHILE(FLG2.NE.1)
        FLG2=1
        RL=RM
        R=RL-DSQRT(MGP/DEN0)*0.5D0
        DR=(RM-R)*2.D0
        RM=R-DR*0.5D0
        MR=PI*DEN0*RM**2
        NUM=IDNINT((MG-MR)/MGP)-PP
        PP=PP+NUM
        MR=(MG-PP*MGP)
        RM=DSQRT(MR/(PI*DEN0))
        DR=RL-RM
        R=RM+DR*0.5D0
        IF(RM.GT.DR*0.5D0) THEN
          FLG2=0
          NRG=NRG+1
          RRNG(NRG)=R
          NRNG(NRG)=NUM
        ENDIF
      ENDDO
C
C   LAST RING.
C-----
      R=RL*0.5D0

```

```

NRG=NRG+1
RRNG (NRG) =R
NRNG (NRG) =NACC-PP+NUM
ELSE
C
C  FIND FIRST RADIUS.
C-----
      R=0.95D0*ARAD
      ROLD=1.0
      FLAG=0
      DOWHILE (DABS (R-ROLD) .GT. 1.D-06 .AND. FLAG.LT.20)
        FLAG=FLAG+1
        DEN=DEN0*DEXP (R*RSINV)
        FN=2.D0*(R-ARAD)+DSQRT (MGP/DEN)
        DF=2.D0-0.5D0*RSINV*DSQRT (MGP/DEN)
        ROLD=R
        R=ROLD-FN/DF
      ENDDO
      DR=(ARAD-R)*2.D0
      RM=R-DR*0.5D0
      MR=DEN0*2.D0*PI*(1.D0-DEXP (RM*RSINV)*(1.D0-RM*RSINV))/RSINV**2
      NUM=IDNINT ((MG-MR)/MGP)
      MR=MG-NUM*MGP
      ROLD=1.D0
      FLAG=0
      DOWHILE (DABS (RM-ROLD) .GT. 1.D-06 .AND. FLAG.LT.10)
        FLAG=FLAG+1
        DEN=DEN0*DEXP (RM*RSINV)
        FN=MR-2.D0*PI*(DEN0-DEN*(1.D0-RM*RSINV))/RSINV**2
        DF=-2.D0*PI*RM*DEN
        ROLD=RM
        RM=ROLD-FN/DF
      ENDDO
      DR=ARAD-RM
      R=RM+DR*0.5D0
      PP=NUM
      RRNG (1) =R
      NRNG (1) =NUM
      NRG=1
C
C  FIND RADII OF RINGS, OUTSIDE IN.
C-----
      FLG2=0
      DOWHILE (FLG2.NE.1)
        FLG2=1
        RL=RM
        R=RL-DR*0.5D0
        ROLD=1.D0

```

```

10      FLAG=0
      DOWHILE (DABS (R-ROLD) .GT. 1.D-06 .AND. FLAG .LT. 20)
          FLAG=FLAG+1
          DEN=DEN0*DEXP (R*RSINV)
          FN=2.D0*(R-RL)+DSQRT (MGP/DEN)
          DF=2.D0-0.5D0*RSINV*DSQRT (MGP/DEN)
          ROLD=R
          R=ROLD-FN/DF
      ENDDO
      IF ( (RL-R) .LT. 1.D-8) THEN
          R=RL-DR
          GOTO 10
      ENDIF
      DR=(RM-R)*2.D0
      RM=R-DR*0.5D0
      DEN=DEN0*DEXP (RM*RSINV)
      MR=2.D0*PI*(DEN0-DEN*(1.D0-RM*RSINV))/RSINV**2
      NUM=IDNINT ( (MG-MR)/MGP)-PP
      PP=PP+NUM
      MR=(MG-PP*MGP)
      ROLD=1.D0
      FLAG=0
      DOWHILE (DABS (RM-ROLD) .GT. 1.D-06 .AND. FLAG .LT. 10)
          FLAG=FLAG+1
          DEN=DEN0*DEXP (RM*RSINV)
          FN=MR-2.D0*PI*(DEN0-DEN*(1.D0-RM*RSINV))/RSINV**2
          DF=-2.D0*PI*RM*DEN
          ROLD=RM
          RM=ROLD-FN/DF
      ENDDO
      DR=RL-RM
      R=RM+DR*0.5D0
      IF (RM.GT.DR*0.5D0) THEN
          FLG2=0
          NRG=NRG+1
          RRNG (NRG) =R
          NRNG (NRG) =NUM
      ENDIF
      ENDDO
C
C  LAST RING:
C -----
      NRG=NRG+1
      R=RL*0.5D0
      RRNG (NRG) =R
      NRNG (NRG) =NACC-PP+NUM
      ENDIF
C

```

C PLACE PARTICLES INTO RINGS.

```

C-----
      IF(NRG.GT.20) CALL TERROR(' ARRAY OVERFLOW IN ACCRET. ')
15  P=ACCLIST(NACC)
      RAD=DSQRT((POS(P,1)-X0)**2+(POS(P,2)-Y0)**2)
      SINA=(POS(P,2)-Y0)/RAD
      COSA=(POS(P,1)-X0)/RAD
      VT=(VEL(P,2)-VY0)*COSA-(VEL(P,1)-VX0)*SINA
      THETA=THOLD+VT*DTIME/RAD
      THOLD=THETA
      NINDX=NACC
      DO I=1,NRG
        R=RRNG(I)
        DEN=DEN0*DEXP(R*RSINV)
        VR=-MDOT/(2.D0*PI*R*DEN)
        VT=DSQRT(MASS1/R+A*R+ISS*GAMMA*R*RSINV*DEN**GMONE)
        DTH=2.D0*PI/DFLOAT(NRNG(I))
        HP=2.D0*ZETA*DSQRT(MGP/DEN)
        DO J=1,NRNG(I)
          P=ACCLIST(NINDX)
          COSA=DCOS(THETA)
          SINA=DSIN(THETA)
          POS(P,1)=R*COSA+X0
          POS(P,2)=R*SINA+Y0
          VEL(P,1)=-VT*SINA+VX0+VR*COSA
          VEL(P,2)=VT*COSA+VY0+VR*SINA
          MASS(P)=MGP
          H(P)=HP
          AMDA=AMDA+VT*R
          THETA=THETA+DTH
          NINDX=NINDX-1
        ENDDO
        AMDA=AMDA+NRNG(I)*VT*R
        THETA=0.5D0*DTH+THETA
      ENDDO

```

C

C SET VELOCITY OF ACCRETED PARTICLES AT TIME STEP DTIME/2.

```

C-----
      DO K=1,NDIM
        DO I=1,NACC
          P=ACCLIST(I)
          VET(P,K)=VEL(P,K)
        ENDDO
      ENDDO

```

C

C ADJUST H OF INNER DISK PARTICLES AT TIME=0

C-----

```

      IF(N.EQ.0) THEN
        CALL MAKETRE
C      -----
        DO I=1,NACC
          P=ACCLIST(I)
          FLAG=0
65        FLAG=FLAG+1
          IF(FLAG.GE.20) THEN
            WRITE(ULOG,*) N,P,POS(P,1),POS(P,2),H(P)
            WRITE(ULOG,*) NACC
            CALL TERROR(' UNABLE TO ADJUST SMOOTHING IN ACCRET. ')
          ENDIF
          CALL WALKIT(P,H(P),NVAL)
C      -----
          IF(NVAL.LT.(8*NNBMAX)/10) THEN
            H(P)=H(P)*1.1D0
            GOTO 65
          ENDIF
          IF(NVAL.GT.NSTMAX) THEN
            WRITE(ULOG,*) ' IN ACCRET NVAL.GT.200 FOR PART. ',P
            WRITE(ULOG,*) N,NVAL,H(P),TNOW
            H(P)=H(P)*0.5D0
            GOTO 65
          ENDIF
          DO J=1,NVAL
            NBDR(J)=WORKVEC(J)
          ENDDO
          IF(NVAL.GT.NNBMAX) THEN
            CALL PKSRT(NVAL,NBDR,INDX)
            H(P)=(NBDR(INDX(NNBMAX))+NBDR(INDX(8*NNBMAX/10)))*0.5D0
          ENDIF
        ENDDO
      ENDIF
      AMDA=AMDA*MGP
25    LT(0)=LT(0)-AAMV(3)
      U(1)=AAM
      TAEK=TAEK+AEK
      TAE=TAE+ACCE

99    RETURN
      END@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
C
C
C
C
C
C
C
C
C*****

```

SUBROUTINE AMFLUX

```

C
C
C   SUBROUTINE TO KEEP TRACK OF THE ANGULAR MOMENTUM FLUX ACROSS
C   THE ANNULI BOUNDARIES AT R=1,2,... . THIS FLUX IS STORED IN
C   THE ARRAY LF AND IS WRITTEN TO A FILE IN SUBROUTINE OUTDISP.
C   RONALD DRIMMEL 11/92
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

      DOUBLE PRECISION  X,Y,R,COSA,SINA,VT
      INTEGER M,P,I,K

      DO K=1,NDIM
        CMVEL(K)=0.D0
        CMPOS(K)=0.D0
      ENDDO
      MTOT=0.D0
      AMVEC(3)=0.D0
      DO I=1,NGP-NAGP
        P=BODLIST(I)
        MTOT=MTOT+MASS(P)
        AMVEC(3)=AMVEC(3)+MASS(P)*(POS(P,1)*VET(P,2)-
&      POS(P,2)*VET(P,1))
      ENDDO

      DO K=1,NDIM
        DO I=1,NGP-NAGP
          P=BODLIST(I)
          CMVEL(K)=CMVEL(K)+MASS(P)*VET(P,K)
          CMPOS(K)=CMPOS(K)+MASS(P)*POS(P,K)
        ENDDO
        CMPOS(K)=CMPOS(K)/MTOT
        CMVEL(K)=CMVEL(K)/MTOT
      ENDDO

C   RESET 'GRID' PARTICLES FOR SPECTRAL TIME ANALYSIS.
C   -----
      DELR=(1.D0-0.1D0)/DFLOAT(NGPTS-1)
      DO I=NBODIES+1,NBODIES+NGPTS
        POS(I,1)=0.1D0+DELR*(I-1-NBODIES)+CMPOS(1)
        POS(I,2)=CMPOS(2)
      ENDDO
      DO I=NBODIES+1+NGPTS,NBODIES+2*NGPTS
        POS(I,1)=-0.1D0-DELR*(I-1-NBODIES-NGPTS)+CMPOS(1)
        POS(I,2)=CMPOS(2)
      ENDDO

```

```

DO I=NBODIES+1+2*NGPTS,NBODIES+3*NGPTS
  POS(I,2)=0.1D0+DELR*(I-1-NBODIES-2*NGPTS)+CMPOS(2)
  POS(I,1)=CMPOS(1)
ENDDO
DO I=NBODIES+1+3*NGPTS,NBODIES+4*NGPTS
  POS(I,2)=-0.1D0-DELR*(I-1-NBODIES-3*NGPTS)+CMPOS(2)
  POS(I,1)=CMPOS(1)
ENDDO

DO I=NMP+1,NGP-NAGP
  P=BODLIST(I)
  X=POS(P,1)-CMPOS(1)
  Y=POS(P,2)-CMPOS(2)
  R=DSQRT(X*X+Y*Y)
  M=INT(R/ANW)+1
  IF(M.LE.50.AND.M.NE.MINDX(P)) THEN
    COSA=X/R
    SINA=Y/R
    VT=(VEL(P,2)-CMVEL(2))*COSA-(VEL(P,1)-CMVEL(1))*SINA
    IF((M-MINDX(P)).GT.0) THEN
      LT(M-1)=LT(M-1)+MASS(P)*VT*DFLOAT(M-1)*ANW
    ELSE
      LT(M)=LT(M)-MASS(P)*VT*DFLOAT(M)*ANW
    ENDIF
    MINDX(P)=M
  ELSEIF(M.GT.50) THEN
    MINDX(P)=51
  ENDIF
ENDDO

RETURN
END

@PROCESS DC(BODYCEL,CONCOM,GAS)
@PROCESS DIRECTIVE(' *VDIR:')
C*****
C
C
C
C
C
C
C
C
C
SUBROUTINE CORRPOS(RC)
C*****
C
C
C
C
C
C
C
C
C
SUBROUTINE TO APPLY A CORRECTION FACTOR TO THE POSITIONS TO
C MAINTAIN SECOND ORDER ACCURACY WHEN OUTPUTTING PARTICLE DATA
C TO BODY DATA FILE OR WHEN COMPUTING ENERGY DIAGNOSTICS. THE
C ARGUMENT RC INDICATES WHETHER THE CORRECTION FACTOR IS TO BE
C APPLIED (CORRECT) OR REMOVED (RESET).

```

```

C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C  DECLARATION OF LOCAL VARIABLES.
C  -----

      CHARACTER*7 RC
      DOUBLE PRECISION DT2,RCSIGN
      INTEGER P,K,I

C=====

      IF(RC.EQ.'CORRECT') THEN
        RCSIGN=-1.
      ELSE
        RCSIGN=1.
      ENDIF

C  LOOP OVER ALL SPATIAL COORDINATES FOR ALL BODIES.
C  -----

C*VDIR: IGNORE RECRDEPS
      DO 200 K=1,NDIM
        DO 100 I=1,NBODLIS
          P=BODLIST(I)
          DT2=(DTS/2.D0**TBIN(P))**2
          POS(P,K)=POS(P,K)+RCSIGN*ACC(P,K)*DT2*0.125D0
100      CONTINUE
200      CONTINUE

      IF(NMP.NE.0) THEN
        DT2=(DTS/2.D0**TBIN(1))**2
        DO K=1,NDIM
C*VDIR: IGNORE RECRDEPS
          DO I=1,NACC
            P=ACCLIST(I)
            POS(P,K)=POS(P,K)+RCSIGN*ACC(1,K)*DT2*0.125D0
          ENDDO
        ENDDO
      ENDIF

      RETURN
      END
@PROCESS DC(BODYCEL,CONCOM,GAS)

```

```

C*****
C
C
C          SUBROUTINE CORRVEL
C
C
C*****
C
C          SUBROUTINE TO ESTIMATE THE VELOCITIES OF THE GAS PARTICLES
C          FOR THE TIMESTEP DTIME/2.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C  DECLARATION OF LOCAL VARIABLES.
C  -----

      INTEGER P,K,I

C=====

C
C  RESETS VELOCITIES AFTER INITIAL POSITION CORRECTION MADE.
C-----
      IF(BNOW.EQ.NCMAX) THEN
        DO K=1,NDIM
          DO I=1,NADVLIS
            P=ADVLIST(I)
            VEL(P,K)=DVEL(P,K)
          ENDDO
        ENDDO
        NCMAX=NCMAX-1
      ENDIF

      RETURN
      END
@PROCESS DC(BODYCEL,CONCOM,GAS)
@PROCESS DIRECTIVE('*VDIR:')
C*****
C
C          SUBROUTINE COURANT(N)
C
C=====
C

```

```

C
C   SUBROUTINE TO CALCULATE THE TIME STEP USING THE COURANT
C   CONDITION, WHERE CN IS THE COURANT NUMBER. THE EXPRESSION
C   USED BELOW IS TAKEN FROM HIOTELIS.
C   N. HIOTELIS AND R. DRIMMEL '91
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C
C   DECLARATION OF LOCAL VARIABLES
C
C-----

      INTEGER I,P,N,BIN,NUM,J
      DOUBLE PRECISION VELI, FORC, LEN, X, A, AA, DTSOLD, R

C-----

      25      NMIN=0
C   FIND BINS THAT NEED POSSIBLE REBINNING.
C-----
      IF (N.NE.0) THEN
          AA=DABS (TELAPS/DTS-IDNINT (TELAPS/DTS) )
          DOWHILE (AA.GT.1.D-06.AND.NMIN.LE.NMAX)
              NMIN=NMIN+1
              AA=TELAPS/ (DTS/2.DO**NMIN)
              AA=DABS (AA-IDNINT (AA) )
          ENDDO

C
C   MAKE LIST OF PARTICLES IN BINS.
C-----
          CALL WHENIGE (NBODIES, TBIN, 1, NMIN, ADVLIST, NADVLIS)
          NGAS=0
          I=0
          DOWHILE (ADVLIST (I+1) .LE. NGP.AND. (I+1) .LE. NADVLIS)
              I=I+1
          ENDDO
          NGAS=I

C
C   CHECK TO SEE IF SMALLEST TIME BIN EMPTIED DUE TO ACCRETION.
C   IF SO, CHANGE NMAX AND MAKE A HALF STEP.
C-----
          IF (NMP.NE.0) THEN
              IF (NADVLIS.EQ.NMP.AND.NMIN.EQ.NMAX) THEN
                  NMAX=NMAX-1

```

```

        DTIME=DTS/2.D0**NMAX
        DTIME2=DTIME*0.5D0
        BNOW=NMAX
        TBIN(NMP)=NMAX
        CALL WHENIEQ(NBODIES,TBIN,1,BNOW,ADVLIST,NADVLIS)
        NGAS=0
        I=0
        DOWHILE(ADVLIST(I+1).LE.NGP.AND.(I+1).LE.NADVLIS)
            I=I+1
        ENDDO
        NGAS=I
        CALL HALFSTP(N)
        CALL AMFLUX
        CALL ACCRET(N)
        GOTO 25
    ENDIF
    ELSEIF(NADVLIS.EQ.0) THEN
        CALL ERROR(' NO PARTICLES FOUND TO BE REBINNED. ')
    ENDIF
ENDIF
C
C  FIND TIME STEPS.
C-----
        NUM=0
        IF(ADVLIST(1).EQ.NMP) NUM=1
        DO I=1+NUM,NGAS
            P=ADVLIST(I)
            FORC=DSQRT(ACC(P,1)**2+ACC(P,2)**2)
            VELI=DSQRT(VEL(P,1)**2+VEL(P,2)**2)
            LEN=MIN(EPS,0.5D0*H(P))
C        LEN=0.5D0*H(P)
            DTIME=CN*MIN(DSQRT(LEN/FORC),LEN/VELI)
            DT(P)=MIN(DT(P),DTIME)
        ENDDO
        DO I=NGAS+1,NADVLIS
            P=ADVLIST(I)
            FORC=DSQRT(ACC(P,1)**2+ACC(P,2)**2)
            VELI=DSQRT(VEL(P,1)**2+VEL(P,2)**2)
            DT(P)=CN*MIN(DSQRT(EPS/FORC),EPS/VELI)
        ENDDO
        DTIME=100000.D0
        DO I=1+NUM,NADVLIS
            P=ADVLIST(I)
            DTIME=MIN(DTIME,DT(P))
        ENDDO
C
C  SET NEW SYSTEM TIME STEP.

```

```

C-----
      IF (NMIN.EQ.0) THEN
        DTSOLD=DTS
        DTS=0.D0
        DO I=1+NUM,NADVLIS
          P=ADVLIS(I)
          DTS=MAX(DTS,DT(P))
        ENDDO
        DTS=0.5D0*DTS
        N=N+1
        TELAPS=0.D0
      ENDIF

C
~  FIND NMAX OF NEXT STEP.
-----
      BIN=INT(LOG(DTS/DTIME)/LOG(2.D0)+1.D0)
      IF (BIN.GE.NMIN) NMAX=BIN
      IF (NMAX.GT.NBINMAX) CALL TERROR(' NMAX GT NBINMAX. ')
      DTIME=DTS/2.D0**NMAX

C
C  SET TIME BINS.
C-----
      IF (NMP.NE.0) TBIN(NMP)=NMAX
      IF (NMIN.NE.0) THEN
        DO I=1+NMP,NADVLIS
          P=ADVLIS(I)
          BIN=INT(LOG(DTS/DT(P))/LOG(2.D0)+1.D0)
          IF (TBIN(P).NE.BIN) THEN
            IF (BIN.GE.NMIN) THEN
              X=2.D0** (TBIN(P)-BIN)
              DTIME2=(DTS/2.D0**TBIN(P))**2
              X=(1.D0-X)*(1.D0+X)*DTIME2/8.D0
              DO K=1,NDIM
                POS(P,K)=POS(P,K)-X*ACC(P,K)
              ENDDO
              TBIN(P)=BIN
            ELSEIF (TBIN(P).GT.NMAX) THEN
              BIN=NMIN
              X=2.D0** (TBIN(P)-BIN)
              DTIME2=(DTS/2.D0**TBIN(P))**2
              X=(1.D0-X)*(1.D0+X)*DTIME2/8.D0
              DO K=1,NDIM
                POS(P,K)=POS(P,K)-X*ACC(P,K)
              ENDDO
              TBIN(P)=BIN
            ENDIF
          ENDIF
        ENDIF
      ENDIF

```

```

      ENDDO
    ELSE
      DO I=1+NMP,NADVLIS
        P=ADVLIST(I)
        BIN=INT(LOG(DTS/DT(P))/LOG(2.D0)+1.D0)
        X=DTSOLD*2.D0**BIN/(DTS*2.D0**TBIN(P))
        DTIME2=(DTSOLD/2.D0**TBIN(P))**2
        X=(1.D0-X)*(1.D0+X)*DTIME2/8.D0
        DO K=1,NDIM
          POS(P,K)=POS(P,K)-X*ACC(P,K)
        ENDDO
        TBIN(P)=BIN
      ENDDO
    ENDIF

C
C  FIND RMIN AND RMAX OF THE TIME BINS THAT HAVE JUST BIN READJUSTED.
C-----
      DO I=NMIN,NMAX
        RMAX(I)=0.D0
        RMIN(I)=100.D0
      ENDDO
      DO I=1+NMP,NGAS
        P=ADVLIST(I)
        R=DSQRT((POS(P,1)-POS(NMP,1))**2+(POS(P,2)-POS(NMP,2))**2)
        RMAX(TBIN(P))=MAX(RMAX(TBIN(P)), (R+H(P)))
        RMIN(TBIN(P))=MIN(RMIN(TBIN(P)), (R-H(P)))
      ENDDO

C  FIND NEW BNOW.
C-----
      BNOW=0
      A=2.D0*TELAPS
      AA=DABS(A/DTS-IDNINT(A/DTS))
      DOWHILE(AA.GT.1.D-06.AND.BNOW.LE.NMAX)
        BNOW=BNOW+1
        AA=A/(DTS/2.D0**BNOW)
        AA=DABS(AA-IDNINT(AA))
      ENDDO

C
C  MAKE LIST OF PARTICLES TO BE ADVANCED IN THE NEXT TIME STEP.
C-----
      CALL WHENIEQ(NBODIES,TBIN,1,BNOW,ADVLIST,NADVLIS)
      IF(NADVLIS.EQ.0) THEN
        WRITE(ULOG,*) '      BNOW=',BNOW
        CALL TERROR(' NO PARTICLES IN THIS TIME BIN. ')
      ENDIF
      NGAS=0

```

```

      I=0
      DOWHILE(ADVLIST(I+1).LE.NGP.AND.(I+1).LE.NADVLIS)
        I=I+1
      ENDDO
      NGAS=I

      DTIME2=0.5D0*DTIME

      RETURN
      END
@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
      SUBROUTINE DENSITY
C
C
C*****
C
C
C      SUBROUTINE TO COMPUTE THE DENSITY, SOFTENED DENSITY, DELV, C,
C      AND Q FOR EACH GAS PARTICLE. ADDAPTED FROM ORIGINAL HYDRO
C      SUBROUTINE WRITTEN BY N. HIOTELIS. R. DRIMMEL 9/92
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C
C  DECLARATION OF LOCAL VARIABLES.
C  -----
      INTEGER NVAL,I,J,K,L,I1,P,NUM
      DOUBLE PRECISION DR2,DR,UH,AA,DL,DX,HIJ,HINV,H2INV,
&      DY,DVX,DVY,APR,FPI,H4INV,DV,RHDV,PS,KERN,HTA,CROSS

C-----
C
C      FIND GAS DENSITY AND VELOCITY DIVERGENCE FOR ACTIVE PARTICLES.
C
C-----
      HTA=0.001D0
      NUM=0
      IF(ADVLIST(1).EQ.NMP) NUM=1
      DO I=1+NUM,NGAS
        P=ADVLIST(I)
        RH(P)=MASS(P)*C1/(H(P)*H(P))

```

```

      F(P)=0.D0
    ENDDO
    DO I=1+NUM,NGAS
      P=ADVLIST(I)
      DV=0.0D0
      DO L=1,NNBS(P)
        J=LIST(L,P)
        HIJ=(H(P)+H(J))*0.5D0
        DX=POS(P,1)-POS(J,1)
        DY=POS(P,2)-POS(J,2)
        DVX=VET(P,1)-VET(J,1)
        DVY=VET(P,2)-VET(J,2)
        HINV=1.D0/HIJ
        UH=DSQRT(DX*DX+DY*DY)*HINV
        H2INV=HINV*HINV
        H4INV=H2INV*H2INV
        APR=DX*DVX+DY*DVY
        CROSS=DVX*DY-DVY*DX
        AA=NINTERP*UH
        I1=AA
        DL=AA-I1
        DR=1.D0-DL
        I1=I1+1
        FPI=(DR*VDIVW3(I1)+DL*VDIVW3(I1+1))*C2*MASS(J)*H4INV
        DV=DV+APR*FPI
        F(P)=F(P)+CROSS*FPI
        RH(P)=RH(P)+C1*(VW3(I1+1)*DL+VW3(I1)*DR)*MASS(J)*H2INV
      ENDDO
    C   FOR GAMMA=1:
    CC   C(P)=DSQRT(ISS)
        C(P)=DSQRT(GAMMA*ISS*RH(P)**GMONE)
    C   WITH FIRST LAW:
    CC   C(P)=DSQRT(GAMMA*GMONE*U(P))
        DV=DV/RH(P)
        F(P)=DABS(DV)/(DABS(DV)+DABS(F(P)/RH(P))+HTA*C(P)/H(P))
        DELV(P)=-DV
    ENDDO
  C
  C   FIND DENSITY AND DIVERGENCE OF VELOCITY FOR ACCRETED PARTICLES.
  C
  C-----
    IF(BNOW.EQ.TBIN(NMP).OR.TBIN(NMP).EQ.-1) THEN
    DO I=1,NACC
      P=ACCLIST(I)
      RH(P)=MASS(P)*C1/(H(P)*H(P))
      F(P)=0.D0
    ENDDO
    DO I=1,NACC

```

```

P=ACCLIST(I)
DV=0.0D0
DO L=1,NNBS(P)
  J=LIST(L,P)
  HIJ=(H(P)+H(J))*0.5D0
  DX=POS(P,1)-POS(J,1)
  DY=POS(P,2)-POS(J,2)
  DVX=VET(P,1)-VET(J,1)
  DVY=VET(P,2)-VET(J,2)
  HINV=1.D0/HIJ
  UH=DSQRT(DX*DX+DY*DY)*HINV
  H2INV=HINV*HINV
  H4INV=H2INV*H2INV
  APR=DX*DVX+DY*DVY
  CROSS=DVX*DY-DVY*DX
  AA=NINTERP*UH
  I1=AA
  DL=AA-I1
  DR=1.D0-DL
  I1=I1+1
  FPI=(DR*VDIVW3(I1)+DL*VDIVW3(I1+1))*C2*MASS(J)*H4INV
  DV=DV+APR*FPI
  F(P)=F(P)+CROSS*FPI
  RH(P)=RH(P)+C1*(VW3(I1+1)*DL+VW3(I1)*DR)*MASS(J)*H2INV
ENDDO
C  FOR POLYTROPIC EOS:
C  FOR GAMMA=1:
CC    C(P)=DSQRT(ISS)
      C(P)=DSQRT(GAMMA*ISS*RH(P)**GMONE)
C  WITH FIRST LAW:
CC    C(P)=DSQRT(GAMMA*GMONE*U(P))
      DV=DV/RH(P)
      F(P)=DABS(DV)/(DABS(DV)+DABS(F(P)/RH(P))+HTA*C(P)/H(P))
      DELV(P)=-DV
ENDDO
ENDIF
C
C  ESTIMATE SMOOTHED DENSITIES
C
C-----
      NVAL=0
      IF(ADVLIST(1).EQ.NMP) NVAL=1
      DO I=1+NVAL,NGAS
        P=ADVLIST(I)
        H2INV=SOF**2/(H(P)*H(P))
        PS=MASS(P)*C1*H2INV*RH(P)
        RHDV=PS*DELV(P)
        AVM(P)=MASS(P)

```

```

NUM=1
DO L=1,NNB(P)
  J=SLIST(L,P)
  HIJ=(H(P)+H(J))*0.5D0
  DX=POS(P,1)-POS(J,1)
  DY=POS(P,2)-POS(J,2)
  HINV=SOF/HIJ
  DR2=DX*DX+DY*DY
  DR=DSQRT(DX*DX+DY*DY)
  UH=DR*HINV
  IF(UH.LT.1.D0) THEN
    AVM(P)=AVM(P)+MASS(J)
    NUM=NUM+1
    H2INV=HINV*HINV
    AA=NINTERP*UH
    I1=AA
    DL=AA-I1
    DR=1.-DL
    I1=I1+1
    KERN=C1*(VW3(I1+1)*DL+VW3(I1)*DR)*H2INV*MASS(J)
    PS=PS+KERN*RH(J)
    RHDV=RHDV+KERN*RH(J)*DELV(J)
  ENDIF
ENDDO
IF(NUM.EQ.1) THEN
  WRITE(ULOG,*) ' NO NEIGHBORS FOR',P
  WRITE(ULOG,*) POS(P,1),POS(P,2),H(P),NNB(P)
ENDIF
AVM(P)=AVM(P)/DFLOAT(NUM)
RHS(P)=PS/RH(P)
DRHDT(P)=RHS(P)*DELV(P)-2.D0*RHDV/RH(P)
ENDDO

```

```

RETURN
END@PROCESS DC (BODYCEL, CONCOM, GAS)

```

```

C*****
C
C
C          SUBROUTINE ENDRUN
C
C
C*****
C
C
C          SUBROUTINE TO END THE SIMULATION.
C
C=====

```

```
/*INCLUDE NEWHEAD FORTRAN A
```

```
C=====
```

```
C  CORRECT POSITIONS, OUTPUT PHASE SPACE DATA TO ASCII FILE.
```

```
C  -----
```

```
C      CALL CORRPOS('CORRECT')
```

```
C      -----
```

```
C      CALL ZEROPOT
```

```
C      -----
```

```
C      CALL GRAVITY('POT ')
```

```
C      -----
```

```
C      CALL OUTASCI
```

```
C      -----
```

```
C  FINISH TIMING, CLOSE DATA FILES.
```

```
C  -----
```

```
C      CALL SECONDS(CPUT1)
```

```
C      -----
```

```
C      CALL OUTCPU
```

```
C      -----
```

```
C      CALL STOPOUT
```

```
C      -----
```

```
      RETURN
```

```
      END
```

```
@PROCESS DC(BODYCEL,CONCOM,GAS)
```

```
C*****
```

```
C
```

```
C
```

```
      SUBROUTINE ENERGY
```

```
C
```

```
C
```

```
C*****
```

```
C
```

```
C
```

```
C      SUBROUTINE TO COMPUTE DIAGNOSTICS FOR THE SYSTEM: TOTAL ENERGY,  
C      TOTAL KINETIC ENERGY, TOTAL POTENTIAL ENERGY, ANGULAR MOMENTUM,  
C      CENTER OF MASS COORDINATES, AND CENTER OF MASS VELOCITY.  THE  
C      LOCAL VARIABLES CMVEL, AND AMVEC ARE CENTER OF MASS  
C      POSITION AND VELOCITY, AND TOTAL ANGULAR MOMENTUM OF THE SYSTEM,  
C      RESPECTIVELY.
```

```
C
```

```
C
```

```
C=====
```

```
/*INCLUDE NEWHEAD FORTRAN A
```

```

C  DECLARATION OF LOCAL VARIABLES.
C  -----

      DOUBLE PRECISION POT,R
      INTEGER P,K,I

C=====

C  ZERO THE ACCUMULATORS FOR SYSTEM DIAGNOSTICS.
C  -----
      MTOT=0.
      EKTOT=0.
      EPTOT=0.
      UTOT=0.

      DO 100 K=1,NDIM
        CMVEL(K)=0.
        CMPOS(K)=0.
100    CONTINUE

      DO I=1,NBODLIS
        P=BODLIST(I)
        MTOT=MTOT+MASS(P)
      ENDDO

      DO 120 K=1,3
        AMVEC(K)=0.
120    CONTINUE

C-----
C  LOOP OVER BODIES TO COMPUTE SYSTEM MASS AND POTENTIAL ENERGY.
C-----

      DO 150 I=1,NBODLIS
        P=BODLIST(I)
        EPTOT=EPTOT+MASS(P)*PHI(P)
150    CONTINUE
      EPTOT=EPTOT*0.5D0

C-----
C  COMPUTE SYSTEM INTERNAL ENERGY OF GAS PARTICLES.
C-----

      DO I=NMP+1,NGP-NAGP
        P=BODLIST(I)
        UTOT=UTOT+MASS(P)*U(P)
      ENDDO

```

```

C-----
C   COMPUTE SYSTEM KINETIC ENERGY, COMPONENTS OF CENTER OF MASS
C   POSITION AND VELOCITY.
C-----

      DO 250 K=1,NDIM
        DO 200 I=1,NBODLIS
          P=BODLIST(I)
          EKTOT=EKTOT+MASS(P)*VEL(P,K)**2
          CMVEL(K)=CMVEL(K)+MASS(P)*VEL(P,K)
          CMPOS(K)=CMPOS(K)+MASS(P)*POS(P,K)
200      CONTINUE
          CMPOS(K)=CMPOS(K)/MTOT
          CMVEL(K)=CMVEL(K)/MTOT
250      CONTINUE
          EKTOT=EKTOT*0.5D0

C   COMPUTE TOTAL SYSTEM ENERGY.
C   -----
          ETOT=EKTOT+EPTOT+UTOT+QTOT

C-----
C   COMPUTE ANGULAR MOMENTUM OF THE SYSTEM.
C-----

C   IF(NDIM.EQ.2) THEN

      DO 300 I=1,NBODLIS
        P=BODLIST(I)
        AMVEC(3)=AMVEC(3)+MASS(P)*(POS(P,1)*VEL(P,2)-
&          POS(P,2)*VEL(P,1))
300      CONTINUE
C   ELSE IF(NDIM.EQ.3) THEN
C
      DO 400 I=1,NBODLIS
        P=BODLIST(I)
        AMVEC(1)=AMVEC(1)+MASS(P)*(POS(P,2)*VEL(P,3)-
C      &          POS(P,3)*VEL(P,2))
C      AMVEC(2)=AMVEC(2)+MASS(P)*(POS(P,3)*VEL(P,1)-
C      &          POS(P,1)*VEL(P,3))
C      AMVEC(3)=AMVEC(3)+MASS(P)*(POS(P,1)*VEL(P,2)-
C      &          POS(P,2)*VEL(P,1))
C400      CONTINUE
C
C   ENDIF

C   WRITE DIAGNOSTICS TO THE LOG FILE.
C   -----

```

```

      CALL OUTENRG
C      -----

      RETURN
      END
@PROCESS DC (BODYCEL, CONCOM, GAS)
C*****
C
C
C      SUBROUTINE ESTVEL
C
C
C*****
C
C      SUBROUTINE TO ESTIMATE THE VELOCITIES OF THE GAS PARTICLES
C      FOR THE TIMESTEP DTIME/2.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C  DECLARATION OF LOCAL VARIABLES.
C  -----

      INTEGER P,K,I

C=====

C  LOOP OVER VELOCITY COMPONENTS FOR GAS BODIES.
C  -----
      DO K=1,NDIM
        DO I=1,NBODLIS
          P=BODLIST(I)
          VET(P,K)=VET(P,K)+DTIME2*ACC(P,K)
        ENDDO
      ENDDO
      IF(BNOW.NE.NMAX) THEN
        DO K=1,NDIM
          DO I=1,NACC
            P=ACCLIST(I)
            VET(P,K)=VET(P,K)+DTIME2*ACC(1,K)
          ENDDO
        ENDDO
      ENDIF

```

```

      RETURN
    END
@PROCESS DC (BODYCEL, CONCOM, GAS)
C*****
C
      SUBROUTINE FINDNBS (P, NVAL, NUM, NBS, FLAG, PATH, STEPS)
C
C-----
C
C
C      SUBROUTINE TO FIX THE SMOOTHING LENGTH OF A PARTICLE.
C      AS IS, THIS SUBROUTINE ADJUSTS H(P) SO THE NUMBER OF
C      NEIGHBORS WITHIN HIJ* IS > NNBMAX/2, AND < 2*NNBMAX.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C
C  DECLARATION OF LOCAL VARIABLES.
C-----

      INTEGER I, P, FLAG, NVAL, J, NUM, NBS, STEPS, PATH(100)

      FLAG=FLAG+1
      CALL WALKIT(P, H(P)/SOF, NVAL)
      NUM=0
      NBS=0
      DO J=1, NVAL
        HIJ= (H(P)+H(TEMPLIS(J)))*0.5D0
        IF (WORKVEC(J).LT.HIJ/SOF) THEN
          NUM=NUM+1
          IF (WORKVEC(J).LT.HIJ) NBS=NBS+1
        ENDIF
      ENDDO
      IF (FLAG.GE.30) THEN
        WRITE(ULOG,*) P, NUM, NBS, BNOW
        DO I=1, STEPS
          WRITE(ULOG,*) '          ', PATH(I)
        ENDDO
        CALL TERROR(' UNABLE TO ADJUST SMOOTHING IN HPFIX. ')
      ENDIF

      RETURN
    END
@PROCESS DC (BODYCEL, CONCOM, GAS)
C*****

```

```

C
C
C          SUBROUTINE GRAVITY(OPTION)
C
C*****
C
C          SUBROUTINE TO COMPUTE GRAVITATIONAL POTENTIAL AND ACCELERATION.
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C  DECLARATION OF LOCAL VARIABLES.
C  -----
C      CHARACTER*4 OPTION
C=====

C          CALL MAKETRE
C          -----
C          CALL ACCGRAV(OPTION)
C          -----

C          RETURN
C          END
@PROCESS DC(BODYCEL, CONCOM, GAS)
C*****
C
C          SUBROUTINE GRAVSUM(P, NTERMS, OPTION)
C
C*****
C
C          SUBROUTINE TO COMPUTE THE MONOPOLE AND QUADRUPOLE CONTRIBUTIONS
C          TO THE POTENTIAL AND ACCELERATION COMPONENTS FOR BODY P.  THE
C          INTERACTION LIST IS CONTAINED IN THE VECTOR ITERMS, WHICH IS
C          EQUIVALENCED TO THE COMMON ARRAY ACTLIST.
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

```

C DECLARATION OF LOCAL VARIABLES.

C -----

INTEGER MAXNTER

PARAMETER (MAXNTER=NWORKVE/9)

CHARACTER*4 OPTION

DOUBLE PRECISION R3INVEF (MAXNTER), RINVEFF (MAXNTER), DRDELDR,

& PMASS, DRDOTDR (MAXNTER), PHSM, DRSM, ACCSM,

& DX (MAXNTER), DY (MAXNTER), DZ (MAXNTER),

& QR5INV (MAXNTER), PHIQUAD (MAXNTER), SDRDOTD,

& R2INVEF (MAXNTER), ACCI, EPSP, EPSI

INTEGER P, I, QINDEX (NBODSMA), QTERMS (NBODSMA), SMINDEX (NBODSMA),

& NTERMS, ITERMS (NBODSMA), NQTERMS

EQUIVALENCE (ITERMS (1), ACTLIST (1)), (QINDEX (1), TEMPLIS (1)),

& (QTERMS (1), ISUBSET (1)), (SMINDEX (1), PARENT (1)),

& (DX (1), WORKVEC (1)), (DY (1), WORKVEC (MAXNTER+1)),

& (DZ (1), WORKVEC (2*MAXNTER+1)), (R3INVEF (1),

& WORKVEC (3*MAXNTER+1)), (RINVEFF (1),

& WORKVEC (4*MAXNTER+1)), (DRDOTDR (1),

& WORKVEC (5*MAXNTER+1)), (QR5INV (1),

& WORKVEC (6*MAXNTER+1)), (PHIQUAD (1),

& WORKVEC (7*MAXNTER+1)), (R2INVEF (1),

& WORKVEC (8*MAXNTER+1))

C=====

IF (NTERMS.GT.MAXNTER)

& CALL TERROR (' ARRAY OVERFLOW IN GRAVSUM ')

C -----

C-----

C COMPUTE MONOPOLE CONTRIBUTION; TEMPORARILY SET MASS OF BODY P TO

C ZERO TO AVOID POSSIBLE SELF-INTERACTION CONTRIBUTION.

C-----

PMASS=MASS (P)

MASS (P)=0.

EPSP=EPS

C LOOP OVER INTERACTION LIST.

C -----

IF (NMP.GT.0) THEN

 NTERMS=NTERMS+1

 ITERMS (NTERMS)=1

```

ENDIF

DO 30 I=1,NTERMS
  DX(I)=POS(P,1)-POS(ITERMS(I),1)
  DY(I)=POS(P,2)-POS(ITERMS(I),2)
  DZ(I)=POS(P,3)-POS(ITERMS(I),3)
  DRDOTDR(I)=DX(I)**2+DY(I)**2
  SDRDOTD=SQRT(DRDOTDR(I))
  RINVEFF(I)=1./(SDRDOTD+TINY)
  R3INVEF(I)=RINVEFF(I)/(DRDOTDR(I)+TINY)
  EPSI=EPS
  DRDELDLDR=SDRDOTD*NINTERP/(EPSP+EPSI)
  SMINDEX(I)=DRDELDLDR
  SMINDEX(I)=MIN(NINTERP,SMINDEX(I))
  DRSM=MIN(ONE,DRDELDLDR-SMINDEX(I))
  PHSM=(1.-DRSM)*PHSMOOT(SMINDEX(I))+
&      DRSM*PHSMOOT(1+SMINDEX(I))
  ACCSM=(1.-DRSM)*ACSMOOT(SMINDEX(I))+
&      DRSM*ACSMOOT(1+SMINDEX(I))
  RINVEFF(I)=PHSM*RINVEFF(I)
  R3INVEF(I)=ACCSM*R3INVEF(I)
30  CONTINUE

IF(OPTION.NE.'ACC ') THEN

  DO 40 I=1,NTERMS
    PHI(P)=PHI(P)-MASS(ITERMS(I))*RINVEFF(I)
40  CONTINUE

ENDIF

IF(OPTION.NE.'POT ') THEN

  DO 50 I=1,NTERMS
    ACCI=MASS(ITERMS(I))*R3INVEF(I)
    ACC(P,1)=ACC(P,1)-DX(I)*ACCI
    ACC(P,2)=ACC(P,2)-DY(I)*ACCI
    ACC(P,3)=ACC(P,3)-DZ(I)*ACCI
C 50  CONTINUE

ENDIF

C  RESET MASS OF BODY P.
C  -----
    MASS(P)=PMASS

C  IF REQUIRED, COMPUTE QUADRUPOLE CONTRIBUTION.
C  -----

```

```

      IF(USEQUAD) THEN

C   FILTER OUT BODIES.
C   -----

      CALL WHENIGT(NTERMS,ITERMS,1,NBODSMA,QINDEX,NQTERMS)

C   COMPUTE QUADRUPOLE INTERACTION FROM CELLS.
C   -----

      DO 60 I=1,NQTERMS
        QTERMS(I)=ITERMS(QINDEX(I))
        R2INVEF(I)=RINVEFF(QINDEX(I))*RINVEFF(QINDEX(I))
        QR5INV(I)=R3INVEF(QINDEX(I))*R2INVEF(I)
        PHIQUAD(I)=-0.5D0*(DX(QINDEX(I))*2*QUAD(QTERMS(I),1)+
&          DY(QINDEX(I))*2*QUAD(QTERMS(I),3)+
&          2.*DX(QINDEX(I))*DY(QINDEX(I))*QUAD(QTERMS(I),2))
&          *QR5INV(I)
60      CONTINUE

      IF(OPTION.NE.'ACC ') THEN

        DO 70 I=1,NQTERMS
          PHI(P)=PHI(P)+PHIQUAD(I)
70      CONTINUE

      ENDIF

      IF(OPTION.NE.'POT ') THEN

C   WARNING: THE BELOW IS ONLY CORRECT FOR TWO DIMENSIONS.
C   -----
        DO 80 I=1,NQTERMS
          PHIQUAD(I)=5.*PHIQUAD(I)*R2INVEF(I)
          ACC(P,1)=ACC(P,1)+DX(QINDEX(I))*PHIQUAD(I)+
&          (DX(QINDEX(I))*QUAD(QTERMS(I),1)+
&          DY(QINDEX(I))*QUAD(QTERMS(I),2))
&          *QR5INV(I)
          ACC(P,2)=ACC(P,2)+DY(QINDEX(I))*PHIQUAD(I)+
&          (DY(QINDEX(I))*QUAD(QTERMS(I),3)+
&          DX(QINDEX(I))*QUAD(QTERMS(I),2))
&          *QR5INV(I)
80      CONTINUE

      ENDIF

    ENDIF
  
```

```

RETURN
END@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
C
C
C
C
C*****
/*INCLUDE NEWHEAD FORTRAN A

      INTEGER I1,I,P,J,K
      DOUBLE PRECISION DX,DY,DR,HINV,UH,H2INV,AA,DL,RAD,VX,VY,
&      RHVR(NGPTS),FPI,DROLD

C  FIND DENSITY AT GRID POINTS AND OUTPUT EVERY TEN STEPS.
C
C-----
      DO I=NBODIES+1,NBODIES+4*NGPTS
        RH(I)=0.D0
      ENDDO
      DO I=1+NMP,NGP-NAGP
        P=BODLIST(I)
        DX=POS(P,1)-CMPOS(1)
        DY=POS(P,2)-CMPOS(2)
        IF(DABS(DY).LT.H(P)) THEN
          IF(DX.GT.POS(NBODIES+1,1)-H(P)) THEN
            DR=10000.D0
            DO J=NBODIES+1,NBODIES+NGPTS
              DROLD=DR
              DR=DSQRT((POS(J,1)-DX)**2+DY**2)
              IF(DR.LT.H(P)) THEN
                HINV=1.D0/H(P)
                UH=DR*HINV
                H2INV=HINV*HINV
                AA=NINTERP*UH
                I1=AA
                DL=AA-I1
                DR=1.D0-DL
                I1=I1+1
                FPI=C1*(VW3(I1+1)*DL+VW3(I1)*DR)*MASS(P)*H2INV
                RH(J)=RH(J)+FPI
              ELSEIF(DR.GT.DROLD) THEN
                GOTO 15
              ENDIF
            ENDDO
          ENDIF
        ENDIF
      ENDDO

```

```

15      CONTINUE
      ELSEIF (DX.LT.POS(NBODIES+NGPTS+1,1)+H(P)) THEN
        DR=10000.D0
        DO J=NBODIES+NGPTS+1,NBODIES+2*NGPTS
          DROLD=DR
          DR=DSQRT((POS(J,1)-DX)**2+DY**2)
          IF (DR.LT.H(P)) THEN
            HINV=1.D0/H(P)
            UH=DR*HINV
            H2INV=HINV*HINV
            AA=NINTERP*UH
            I1=AA
            DL=AA-I1
            DR=1.D0-DL
            I1=I1+1
            FPI=C1*(VW3(I1+1)*DL+VW3(I1)*DR)*MASS(P)*H2INV
            RH(J)=RH(J)+FPI
          ELSEIF (DR.GT.DROLD) THEN
            GOTO 25
          ENDIF
        ENDDO
25      CONTINUE
      ENDIF
      ELSEIF (DABS(DX).LT.H(P)) THEN
        IF (DY.GT.POS(NBODIES+2*NGPTS+1,2)-H(P)) THEN
          DR=10000.D0
          DO J=NBODIES+2*NGPTS+1,NBODIES+3*NGPTS
            DROLD=DR
            DR=DSQRT((POS(J,2)-DY)**2+DX**2)
            IF (DR.LT.H(P)) THEN
              HINV=1.D0/H(P)
              UH=DR*HINV
              H2INV=HINV*HINV
              AA=NINTERP*UH
              I1=AA
              DL=AA-I1
              DR=1.D0-DL
              I1=I1+1
              FPI=C1*(VW3(I1+1)*DL+VW3(I1)*DR)*MASS(P)*H2INV
              RH(J)=RH(J)+FPI
            ELSEIF (DR.GT.DROLD) THEN
              GOTO 35
            ENDIF
          ENDDO
35      CONTINUE
      ELSEIF (DY.LT.POS(NBODIES+3*NGPTS+1,1)+H(P)) THEN
        DR=10000.D0
        DO J=NBODIES+3*NGPTS+1,NBODIES+4*NGPTS

```

```

DROLD=DR
DR=DSQRT((POS(J,2)-DY)**2+DX**2)
IF(DR.LT.H(P)) THEN
  HINV=1.D0/H(P)
  UH=DR*HINV
  H2INV=HINV*HINV
  AA=NINTERP*UH
  I1=AA
  DL=AA-I1
  DR=1.D0-DL
  I1=I1+1
  FPI=C1*(VW3(I1+1)*DL+VW3(I1)*DR)*MASS(P)*H2INV
  RH(J)=RH(J)+FPI
ELSEIF(DR.GT.DROLD) THEN
  GOTO 45
ENDIF
ENDDO
45  CONTINUE
ENDIF
ENDIF
ENDDO
WRITE(21,*) NGPTS,TNOW,MTOT
DO I=NBODIES+1,NBODIES+NGPTS
  WRITE(21,100) RH(I),RH(I+NGPTS),RH(I+2*NGPTS),RH(I+3*NGPTS)
ENDDO
100  FORMAT(1X,3(1PE16.9,2X),1PE16.9)

RETURN
END
PROCESS DC(BODYCEL,CONCOM,GAS)
PROCESS DIRECTIVE('*VDIR:')
*****

SUBROUTINE HACKCEL

*****

SUBROUTINE TO COMPUTE MASSES, CENTER OF MASS COORDINATES,
AND OPTIONAL QUADRUPOLE MOMENTS OF CELLS, PROCESSING CELLS
IN ORDER OF INCREASING SIZE. THE PERMUTATION VECTOR IS
RETURNED IN THE COMMON VARIABLE CELLIST. VECTORIZATION IS
ACHIEVED BY SIMULTANEOUSLY PROCESSING ALL CELLS AT THE
SAME LEVEL IN THE HIERARCHY (SEE HERNQUIST, J. COMPUT. PHYS.,
SUBMITTED [1988]).

```

```

C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C  DECLARATION OF LOCAL VARIABLES.
C  -----

      INTEGER P, FCELL, LCELL, I, J, K, L, M, N, NSUBC, NNODES

C=====

C  GENERATE PERMUTATION OF CELLS, ACCORDING TO CELLSIZ.
C  -----

      DO 5 I=1, INCELLS
        CELLIST(I)=NBODSMA+INCELLS-(I-1)
5      CONTINUE

C  INITIALIZE PROPERTIES OF CELLS.
C  -----

      DO 10 P=NBODSMA+1, NBODSMA+INCELLS
        MASS(P)=0.
        POS(P,1)=0.
        POS(P,2)=0.
C        POS(P,3)=0.
10     CONTINUE

      IF(USEQUAD) THEN
        DO 30 K=1, 2*NDIM-1
          DO 20 P=NBODSMA+1, NBODSMA+INCELLS
            QUAD(P,K)=0.
20          CONTINUE
30          CONTINUE
        ENDIF

C  PROCESS CELLS IN ORDER OF INCREASING SIZE.
C  -----

      FCELL=1

40     CONTINUE

      IF(FCELL.LE.INCELLS) THEN

C  DETERMINE WHICH CELLS TO PROCESS.

```

```

C -----
C*VDIR: PREFER SCALAR
      DO 50 I=FCELL,INCELLS
        IF (ABS (CELLSIZ (CELLIST (I)) -CELLSIZ (CELLIST (FCELL)))
          &      .LT.0.01*CELLSIZ (CELLIST (FCELL))) THEN

          LCELL=I
        ELSE
          GO TO 60
        ENDIF
50      CONTINUE

60      CONTINUE

C  COMPUTE PROPERTIES OF THE SELECTED CELLS, LOOPING OVER SUBCELLS.
C  -----

      DO 110 J=1,NSUBCEL

        DO 70 I=FCELL,LCELL
          ASUBP (I-FCELL+1)=SUBP (CELLIST (I) ,J)
70      CONTINUE

        CALL WHENIGT (LCELL-FCELL+1,ASUBP,1,0,ISUBSET,NNODES)

        DO I=1,NNODES
          PARENT (I)=CELLIST (ISUBSET (I) +FCELL-1)
          ASUBP (I)=SUBP (PARENT (I) ,J)
        ENDDO

C*VDIR: IGNORE RECRDEPS
      DO 80 I=1,NNODES
        P=PARENT (I)
        K=ASUBP (I)
        MASS (P)=MASS (P) +MASS (K)
        POS (P,1)=POS (P,1) +MASS (K) *POS (K,1)
        POS (P,2)=POS (P,2) +MASS (K) *POS (K,2)
        POS (P,3)=POS (P,3) +MASS (K) *POS (K,3)
C      80      CONTINUE

110      CONTINUE

C*VDIR: IGNORE RECRDEPS
      DO 120 I=FCELL,LCELL
        P=CELLIST (I)
        POS (P,1)=POS (P,1) /MASS (P)

```

```

      POS(P,2)=POS(P,2)/MASS(P)
C      POS(P,3)=POS(P,3)/MASS(P)
120      CONTINUE

C      COMPUTE OPTIONAL QUADRUPOLE MOMENTS.
C      -----

      IF(USEQUAD) THEN

        DO 210 J=1,NSUBCEL

          DO 130 I=FCELL,LCELL
            ASUBP(I-FCELL+1)=SUBP(CELLIST(I),J)
130          CONTINUE

          CALL WHENIGT(LCELL-FCELL+1,ASUBP,1,0,ISUBSET,NNODES)

          DO 140 I=1,NNODES
            PARENT(I)=CELLIST(ISUBSET(I)+FCELL-1)
            ASUBP(I)=SUBP(PARENT(I),J)
140          CONTINUE

          CALL WHENIGT(NNODES,ASUBP,1,NBODSMA,ISUBSET,NSUBC)

          DO 200 M=1,MIN(2,NDIM)
            DO 190 N=M,NDIM

              L=(M-1)*(NDIM-1)+N

              DO 150 I=1,NNODES
                QUAD(PARENT(I),L)=QUAD(PARENT(I),L)+
&                MASS(ASUBP(I))*(3.*(POS(ASUBP(I),M)-
&                POS(PARENT(I),M))*(POS(ASUBP(I),N)-
&                POS(PARENT(I),N)))
150              CONTINUE

              IF(M.EQ.N) THEN
                DO 170 K=1,NDIM
                  DO 160 I=1,NNODES
                    QUAD(PARENT(I),L)=QUAD(PARENT(I),L)-
&                    MASS(ASUBP(I))*(POS(ASUBP(I),K)-
&                    POS(PARENT(I),K))**2
160                  CONTINUE
170                CONTINUE
              ENDIF

              DO 180 I=1,NSUBC
                TEMPLIS(I)=PARENT(ISUBSET(I))

```

```

      QUAD(TEMPLIS(I),L)=QUAD(TEMPLIS(I),L)+
&      QUAD(ASUBP(ISUBSET(I)),L)
180      CONTINUE

190      CONTINUE
200      CONTINUE

210      CONTINUE

      ENDIF

      FCELL=LCELL+1

      GO TO 40

      ENDIF

      RETURN
      END
@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
C      SUBROUTINE HADJUST
C
C
C*****
C
C
C      SUBROUTINE TO DYNAMICALLY ADJUST THE SOFTENING LENGTH H.
C
C
C=====
/*INCLUDE NEWHEAD FORTRAN A
C
C      DEFINITIONS OF LOCAL VARIABLES
C
C=====
      INTEGER I,P,NUM
      DOUBLE PRECISION DTNOW
C
C      CALCULATE NEW H
C-----
      NUM=0
      DTNOW=DTS/2.D0**BNOW
      IF(ADVLIST(1).EQ.NMP) NUM=1

```

```

DO I=1+NUM,NGAS
P=ADVLIST(I)
RHS(P)=RHS(P)+DTNOW*DRHDT(P)
H(P)=2.D0*ZETA*DSQRT(AVM(P)/RHS(P))
ENDDO

RETURN
END

@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
C          SUBROUTINE HALFSTP(N)
C
C
C*****
C
C
C    ADVANCES SYSTEM A HALF STEP.
C
C
C=====
C
/*INCLUDE NEWHEAD FORTRAN A
C
C    ZERO OUT ACCELERATION, COMPUTE ACCELERATION, ADVANCE VELOCITIES.
C    -----
C
C        CALL CORRVEL
C        -----
C        CALL ZEROACC
C        -----
C        CALL GRAVITY('ACC ')
C        -----
C        CALL MAKESAP
C        -----
C        CALL MAKELIS
C        -----
C        CALL DENSITY
C        -----
C        CALL HYDRO(N)
C        -----
C        CALL HADJUST
C        -----
C        CALL STEPVEL
C        -----
C        CALL STEPPOS
C        -----

```

```

      CALL ESTVEL
C      -----

      RETURN
      END
@PROCESS DC (BODYCEL, CONCOM, GAS)
C*****
C
      SUBROUTINE HPFIX (P, NVAL, NUM, NBS, FLAG)
C
C-----
C
C
C      SUBROUTINE TO FIX THE SMOOTHING LENGTH OF A PARTICLE.
C      AS IS, THIS SUBROUTINE ADJUSTS H(P) SO THE NUMBER OF
C      NEIGHBORS WITHIN HIJ* IS > NNBMAX/2, AND < 2*NNBMAX.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C
C  DECLARATION OF LOCAL VARIABLES.
C-----

      DOUBLE PRECISION DH
      INTEGER I, P, FLAG, NVAL, J, NUM, NBS, NBMIN, NBSMAX, PATH(100), STEPS,
&      NBOLD
C-----
C
C  ADJUST H(I)
C-----

      FLAG=0
      STEPS=0
      NBS=0
      NBMIN=12
      NBSMAX=5*NNBMAX/2
45      STEPS=STEPS+1
      PATH(STEPS)=45
      IF (NBS.LT.NBMIN.AND.NUM.LE.NBSMAX) THEN
          DH=0.2D0*H(P)
          GOTO 25
      ELSEIF (NUM.GT.NBSMAX.AND.NBS.GE.NBMIN) THEN
          DH=-0.2D0*H(P)
          GOTO 15
      ELSE
          NBMIN=NBMIN-2

```

```

      GOTO 45
    ENDIF
15    STEPS=STEPS+1
      PATH(STEPS)=15
      H(P)=H(P)+DH
      CALL FINDNBS(P,NVAL,NUM,NBS,FLAG,PATH,STEPS)
      IF(NUM.GT.NBSMAX) THEN
        IF(NBS.LT.NBMIN) NBMIN=NBS
        GOTO 15
      ELSEIF(NBS.LT.NBMIN) THEN
        DH=-0.5D0*DH
        GOTO 35
      ENDIF
      RETURN
25    STEPS=STEPS+1
      PATH(STEPS)=25
      H(P)=H(P)+DH
      NBOLD=NBS
      CALL FINDNBS(P,NVAL,NUM,NBS,FLAG,PATH,STEPS)
      IF(NBS.LT.NBOLD) THEN
        H(P)=H(P)-DH
        NBMIN=NBOLD
        GOTO 99
      ENDIF
      IF(NBS.LT.NBMIN.AND.NUM.LE.NBSMAX) THEN
        DH=0.2D0*H(P)
        GOTO 25
      ELSEIF(NUM.GT.NBSMAX.OR.NBS.GT.3*NNBMAX/2) THEN
        DH=-0.5D0*DH
        GOTO 35
      ENDIF
      RETURN
35    STEPS=STEPS+1
      PATH(STEPS)=35
55    H(P)=H(P)+DH
      CALL FINDNBS(P,NVAL,NUM,NBS,FLAG,PATH,STEPS)
      IF(NUM.GT.NBSMAX.OR.NBS.GT.3*NNBMAX/2) THEN
        DH=-0.5D0*DABS(DH)
        GOTO 35
      ENDIF
      IF(NBS.LT.NBMIN) NBMIN=NBS
99    RETURN
      END
@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
      SUBROUTINE HYDRO(N)
C

```

```

C-----
C
C
C   SUBROUTINE TO CALCULATE THE DENSITY AND PRESSURE-VISCOUS
C   FORCES FOR GASEOUS PARTICLES. THE SMOOTHING LENGTH FOR
C   GASEOUS PARTICLES IS LOCALLY REAJUSTABLE.
C   N. HIOTELIS & R. DRIMMEL '91
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C
C   DECLARATION OF LOCAL VARIABLES.
C-----
      INTEGER I,J,L,I1,FLAG,INDX,P,NUM,NVAL,INDEX(2*NNBMAX),
&      NNBI,K,N
      DOUBLE PRECISION H2,DR2,DR,UH,AA,DL,DX,CIJ,HIJ,HINV,
&      DY,DVX,DVY,APR,PIIJ,PRF,SIJ,FPI,CC,H4INV,HTA,X,Y,
&      RHIJ,MUIJ,VISCX,VISCY,MUMAX,VREL,DUDT1,DUDT2,DTNOW
      DIMENSION INDX(NSTMAX)

C-----
C
C   MAKE PRESSURE AND VISCOSITY ADJUSTMENT TO THE ACCELERATION
C-----
      DTNOW=DTS/2.D0**BNOW
      HTA=0.0025D0
      NUM=0
      IF(ADVLIST(1).EQ.NMP) NUM=1
      DO I=1+NUM,NGAS
         P=ADVLIST(I)
         MUMAX=0.D0
         CC=RH(P)
         VISCX=0.D0
         VISCY=0.D0
         DUDT1=0.D0
         DUDT2=0.D0
         NNBI=NNBS(P)
         DO L=1,NNBI
            J=LIST(L,P)
            HIJ=(H(P)+H(J))*0.5D0
            HINV=1.D0/HIJ
            H4INV=HINV**4
            DVX=VET(P,1)-VET(J,1)
            DVY=VET(P,2)-VET(J,2)

```

```

VREL=DSQRT(DVX*DVX+DVY*DVY)
DX=POS(P,1)-POS(J,1)
DY=POS(P,2)-POS(J,2)
APR=DX*DVX+DY*DVY
DR2=DX*DX+DY*DY
DR=DSQRT(DR2)
UH=DR*HINV
AA=NINTERP*UH
I1=AA
DL=AA-I1
DR=1.D0-DL
I1=I1+1
FPI=(DR*VDIVW3(I1)+DL*VDIVW3(I1+1))*C2*MASS(J)*H4INV
C  POLYTROPIC EQUATION OF STATE:
    IF(GAMMA.NE.2.D0) THEN
        PRF=2.D0*ISS*DSQRT((CC*RH(J))*GMTWO)
    ELSE
        PRF=2.D0*ISS
    ENDIF
C  IDEAL GAS:
C      PRF=2.D0*GMONE*DSQRT(U(P)*U(J)/(CC*RH(J)))
C  PRESSURELESS:
C      PRF=0.D0
      CIJ=(C(P)+C(J))*0.5D0
      HIJ=(H(P)+H(J))*0.5D0
      RHIJ=(CC+RH(J))*0.5D0
      MUIJ=0.5D0*HIJ*APR/(DR2+HTA*HIJ**2)
      PIIJ=-ALFA*CIJ*MUIJ/RHIJ
      IF(SWITCH) MUIJ=MUIJ*(F(P)+F(J))*0.5D0
      MUIJ=MIN(0.D0,MUIJ)
      MUMAX=MAX(DABS(MUIJ),MUMAX)
C      PIIJ=(-ALFA*CIJ+BHTA*MUIJ)*MUIJ/RHIJ
      PIIJ=PIIJ+BHTA*MUIJ**2/RHIJ
      DUDT1=DUDT1+0.5D0*PRF*APR*FPI
      DUDT2=DUDT2+0.5D0*PIIJ*APR*FPI
      SIJ=(PRF+PIIJ)*FPI
      VISCX=VISCX+DX*SIJ
      VISCY=VISCY+DY*SIJ
    ENDDO
    ACC(P,1)=ACC(P,1)-VISCX
    ACC(P,2)=ACC(P,2)-VISCY
C  THE NEXT FIVE STATEMENTS ARE FOR THE POLYTROPIC EQ. OF STATE.
    IF(DTNOW*DUDT1+U(P).LT.0.D0) THEN
        DUDT1=-ISS*CC**GMTWO*DELV(P)
    ENDIF
    UDOT(P)=DUDT1
    IF(N.NE.0) QTOT=QTOT+DUDT2*MASS(P)*DTNOW
C  THE NEXT STATEMENTS ARE TO BE USED WITH THE FIRST LAW.

```

```

C          IF (DTNOW*(DUDT1+DUDT2)+U(P).LT.0.D0) THEN
C              DUDT1=-GMONE*U(P)*DELV(P)/CC
C          ENDIF
C          UDOT(P)=DUDT1+DUDT2
C          FPI=DABS(DELV(P))
      &          +(C(P)+1.2D0*(ALFA*C(P)+BHATA*MUMAX))/(0.5D0*H(P))
C          DT(P)=CN/FPI
C      ENDDO
C      RETURN
C      END
@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
C          SUBROUTINE INBODS
C
C
C*****
C
C      SUBROUTINE TO READ IN THE DATA ASSOCIATED WITH THE BODIES FROM
C      THE ASCII INPUT FILE.  THE RECORDS ARE ASSUMED TO BE IN THE
C      FOLLOWING ORDER:
C
C          INBODIES,NDIM,TNOW,NGP,NMP,QTOT
C          X(1),Y(1),Z(1)
C          VEL(1,1),VEL(1,2),VEL(1,3)
C          MASS(1),U(1),H(1)
C          .
C          .
C          .
C          X(NGP),Y(NGP),Z(NGP)
C          VEL(NGP,1),VEL(NGP,2),VEL(NGP,3)
C          MASS(NGP),U(NGP),H(NGP)
C          X(NGP+1),Y(NGP+1),Z(NGP+1)
C          VEL(NGP+1,1),VEL(NGP+1,2),VEL(NGP+1,3)
C          MASS(NGP+1)
C          .
C          .
C          .
C          X(NBODIES),Y(NBODIES),Z(NBODIES)
C          VEL(NBODIES,1),VEL(NBODIES,2),VEL(NBODIES,3)
C          MASS(NBODIES)
C=====

```

```

/*INCLUDE NEWHEAD FORTRAN A

```

C DECLARATION OF LOCAL VARIABLES.

C -----

INTEGER P,NDIMI,I

C=====

C READ IN BODY DATA.

C -----

READ(UBODSIN,*) NBODIES,NDIMI,TNOW,NGP,NMP,QTOT

IF(NBODIES+4*NGPTS.GT.NBODSMA.OR.NDIMI.NE.NDIM)

& CALL ERROR(' ERROR IN INBODS--INCONSISTENT INPUTS ')

C -----

DO I=1,NGP

READ(UBODSIN,*) POS(I,1),POS(I,2)

READ(UBODSIN,*) VEL(I,1),VEL(I,2)

READ(UBODSIN,*) MASS(I),U(I),H(I)

C READ(UBODSIN,*) MASS(I),H(I),U(I)

ENDDO

DO I=NGP+1,NBODIES

READ(UBODSIN,*) POS(I,1),POS(I,2)

READ(UBODSIN,*) VEL(I,1),VEL(I,2)

READ(UBODSIN,*) MASS(I),H(I)

ENDDO

CLOSE(UBODSIN)

RETURN

END

@PROCESS DC(BODYCEL,CONCOM,GAS)

C*****

C

C

SUBROUTINE INITLIS

C

C

C*****

C

C

C SUBROUTINE TO FIND THE INITIAL LIST OF NEIGHBORS.

C

C

C=====

```
/*INCLUDE NEWHEAD FORTRAN A
```

```
      INTEGER NVAL, I, J, K, L, P, M, NUM, FLAG, INDX(1000), NUMAX, PMAX, NBS,
&      TLIST(2*NNBMAX, NGPMAX), COUNT2, COUNT1, CLIST(20*NNBMAX),
&      NADJ, NFIX, NBSMIN
      DOUBLE PRECISION DR, HIJ, HP, HOLD, RAD(3*NNBMAX), R, AVADJ
```

```
C
```

```
C
```

```
C      FIND NEAREST NEIGHBORS OF ACTIVE PARTICLES
```

```
C-----
```

```
      DO I=NMP+1, NGP-NAGP
        P=BODLIST(I)
        INDL(P)=0
      ENDDO
      DO I=1, NACC
        P=ACCLIST(I)
        INDL(P)=0
      ENDDO
      DO I=1, 4*NNBMAX
        CLIST(I)=0
      ENDDO
      NUMAX=0
      COUNT1=0
      COUNT2=0
      NFIX=0
      AVADJ=0.D0
      DO I=1, NNBSLIS
        P=NBSLIST(I)
        FLAG=0
        NBSMIN=10
        CALL WALKIT(P, (H(P)/SOF), NVAL)
15      NUM=0
        NBS=0
        DO J=1, NVAL
          K=TEMPLIS(J)
          HIJ=(H(K)+H(P))*0.5D0
          IF (WORKVEC(J).LE.HIJ/SOF) THEN
            NUM=NUM+1
            INDX(NUM)=J
            IF (WORKVEC(J).LE.HIJ) THEN
              NBS=NBS+1
            ENDIF
          ENDIF
        ENDDO
        IF (NUM.GT.NUMAX) THEN
          PMAX=P
          NUMAX=NUM
```

```

      HP=H(P)
      L=NVAL
ENDIF
IF (NUM.GT.5*NNBMAX/2.OR.NBS.LT.NBSMIN) THEN
  IF (FLAG.EQ.0) THEN
    FLAG=1
    HOLD=H(P)
    CALL HPFIX(P,NVAL,NUM,NBS,NADJ)
    NFIX=NFIX+1
    AVADJ=AVADJ+NADJ
    IF (H(P).GT.HOLD) THEN
      COUNT1=COUNT1+1
      CLIST(COUNT1)=P
    ELSE
      COUNT2=COUNT2+1
      CLIST(18*NNBMAX+COUNT2)=P
    ENDIF
    IF (NBS.LT.NBSMIN) NBSMIN=NBS
    GOTO 15
  ELSE
    WRITE(ULOG,*) '   NUM=',NUM,'   NVAL=',NVAL,'   NBS=',NBS
    CALL TERROR(' ERROR IN INITLIS. ')
  ENDIF
ENDIF
NNB(P)=NUM
DO J=1,NUM
  RAD(J)=WORKVEC(INDX(J))
  SLIST(J,P)=TEMPLIS(INDX(J))
ENDDO
DO J=1,NUM
  K=TEMPLIS(INDX(J))
  IF (RAD(J).GT.(H(K)/SOF)) THEN
    INDL(K)=INDL(K)+1
    TLIST(INDL(K),K)=P
  ENDIF
ENDDO
ENDDO
C
C   FIND NEAREST NEIGHBORS TO ACCRETED PARTICLES
C
C-----
DO I=1,NACC
  P=ACCLIST(I)
  NBSMIN=10
  FLAG=0
  CALL WALKIT(P,(H(P)/SOF),NVAL)
25  NUM=0
    NBS=0

```

```

DO J=1,NVAL
  HIJ=(H(P)+H(TEMPLIS(J)))*0.5D0
  IF(WORKVEC(J).LT.HIJ/SOF) THEN
    NUM=NUM+1
    INDX(NUM)=J
    IF(WORKVEC(J).LT.HIJ) THEN
      NBS=NBS+1
    ENDIF
  ENDIF
ENDDO
IF(NUM.GT.NUMAX) THEN
  PMAX=P
  NUMAX=NUM
  HP=H(P)
  L=NVAL
ENDIF
IF(NUM.GT.5*NNBMAX/2.OR.NBS.LT.NBSMIN) THEN
  IF(FLAG.EQ.0) THEN
    FLAG=1
    HOLD=H(P)
    CALL HPFIX(P,NVAL,NUM,NBS,NADJ)
    NFIX=NFIX+1
    AVADJ=AVADJ+NADJ
    IF(H(P).GT.HOLD) THEN
      COUNT1=COUNT1+1
      CLIST(COUNT1)=P
    ELSE
      COUNT2=COUNT2+1
      CLIST(18*NNBMAX+COUNT2)=P
    ENDIF
    IF(NBS.LT.NBSMIN) NBSMIN=NBS
    GOTO 25
  ELSE
    WRITE(ULOG,*) '      NUM=',NUM,'      NVAL=',NVAL
    CALL ERROR(' ERROR IN INITLIS ')
  ENDIF
ENDIF
DO J=1,NUM
  K=TEMPLIS(INDX(J))
  IF(WORKVEC(INDX(J)).GT.(H(K)/SOF)) THEN
    INDL(K)=INDL(K)+1
    TLIST(INDL(K),K)=P
  ENDIF
ENDDO
NVAL=0
DO J=1,NUM
  K=TEMPLIS(INDX(J))
  IF(WORKVEC(INDX(J)).LT.(H(P)+H(K))*0.5D0) THEN

```

```

        NVAL=NVAL+1
        SLIST(NVAL,P)=K
    ENDIF
ENDDO
NNB(P)=NVAL
ENDDO
IF(NFIX.NE.0) AVADJ=AVADJ/NFIX
IF(NUMAX.GE.5*NNBMAX) THEN
    WRITE(ULOG,*) ' BAD H:'
    WRITE(ULOG,*) PMAX,L,NUMAX,NNB(PMAX),TNOW
    WRITE(ULOG,100) POS(PMAX,1),POS(PMAX,2),HP,RHS(PMAX)
100    FORMAT(5X,4(1PE12.5,2X))
ENDIF
WRITE(22,200) NMAX,BNOW,NADVLIS,NNBSLIS,COUNT1,COUNT2,NFIX,AVADJ
200    FORMAT(5X,7(I5,2X),1PE12.5)
C
C CHECK FOR OVERFLOW.
C-----
    NUM=0
    DO I=1,NNBSLIS
        P=NBSLIST(I)
        IF(INDL(P).GT.2*NNBMAX) THEN
            NUM=NUM+1
            INDX(NUM)=P
        ENDIF
    ENDDO
    DO I=1,NUM
        P=INDX(I)
        WRITE(ULOG,*) P,INDL(P),POS(P,1),POS(P,2)
    ENDDO
    IF(NUM.NE.0) CALL ERROR(' OVERFLOW IN TLIST ')
    NUM=0
    DO I=1,NACC
        P=ACCLIST(I)
        IF(INDL(P).GT.2*NNBMAX) THEN
            NUM=NUM+1
            INDX(NUM)=P
        ENDIF
    ENDDO
    DO I=1,NUM
        P=INDX(I)
        WRITE(ULOG,*) P,INDL(P),POS(P,1),POS(P,2)
    ENDDO
    IF(NUM.NE.0) CALL ERROR(' OVERFLOW IN TLIST. ')
    IF(COUNT1.GT.18*NNBMAX.OR.COUNT2.GT.2*NNBMAX) THEN
        WRITE(ULOG,*) COUNT1,COUNT2
        CALL ERROR(' OVERFLOW IN CLIST. ')
    ENDIF

```

```

NUM=0
DO I=1,NNBSLIS
  P=NBSLIST(I)
  IF (NNB(P)+INDL(P).GT.7*NNBMAX/2) THEN
    NUM=NUM+1
    INDX(NUM)=P
  ENDIF
ENDDO
IF (NUM.NE.0) THEN
  DO I=1,NUM
    P=INDX(I)
    WRITE(ULOG,300) P,NNB(P),INDL(P),POS(P,1),POS(P,2),TNOW
300    FORMAT(1X,I5,2X,2(I3,2X),3(1PE12.5,2X))
  ENDDO
  CALL TERROR(' OVERFLOW IN SLIST. ')
ENDIF
NUM=0
DO I=1,NACC
  P=ACCLIST(I)
  IF (NNB(P)+INDL(P).GT.7*NNBMAX/2) THEN
    NUM=NUM+1
    INDX(NUM)=P
  ENDIF
ENDDO
IF (NUM.NE.0) THEN
  DO I=1,NUM
    P=INDX(I)
    WRITE(ULOG,300) P,NNB(P),INDL(P),POS(P,1),POS(P,2),TNOW
  ENDDO
  CALL TERROR(' OVERFLOW IN SLIST  ')
ENDIF
CC  GOTO 35
C
C  TAKE EXTRA PARTICLES OUT OF TLISTS OF PARTICLES WHOSE H INCREASED.
C-----

DO I=1,COUNT1
  P=CLIST(I)
  NUM=0
  DO J=1,INDL(P)
    DO 50 K=1,NNB(P)
      IF (TLIST(J,P).EQ.SLIST(K,P)) THEN
        NUM=NUM+1
        INDX(NUM)=J
      ENDIF
50    CONTINUE
  ENDDO
  IF (NUM.GT.0) THEN

```

```

      L=1
      M=INDX(1)-1
      DO J=INDX(1),INDX(NUM)
        IF(J.EQ.INDX(L+1)) THEN
          L=L+1
        ELSE
          M=M+1
          TLIST(M,P)=TLIST(J,P)
        ENDIF
      ENDDO
      DO J=INDX(NUM)+1,INDL(P)
        M=M+1
        TLIST(M,P)=TLIST(J,P)
      ENDDO
      INDL(P)=M
    ENDIF
    CLIST(I)=NUM
    DO J=1,NNB(P)
      K=SLIST(J,P)
      R=DSQRT((POS(P,1)-POS(K,1))**2+(POS(P,2)-POS(K,2))**2)
      IF(R.LT.H(K)/SOF) THEN
        DO M=1,NNB(K)
          IF(SLIST(M,K).EQ.P) GOTO 70
        ENDDO
        NNB(K)=NNB(K)+1
        SLIST(NNB(K),K)=P
70      CONTINUE
        ENDIF
      ENDDO
    ENDDO
C
C  REMOVE PARTICLES IN TLIST THAT NO LONGER FIT CRITERION.
C-----

    DO I=1,COUNT2
      P=CLIST(18*NNBMAX+I)
      HP=H(P)
      NUM=0
      DO J=1,INDL(P)
        K=TLIST(J,P)
        R=DSQRT((POS(P,1)-POS(K,1))**2+(POS(P,2)-POS(K,2))**2)
        HIJ=(HP+H(K))*0.5D0
        IF(R.GT.HIJ/SOF) THEN
          NUM=NUM+1
          INDX(NUM)=J
        ENDIF
      ENDDO
      IF(NUM.GT.0) THEN

```

```

      L=1
      M=INDX(1)-1
      DO J=INDX(1),INDX(NUM)
        IF(J.EQ.INDX(L+1)) THEN
          L=L+1
        ELSE
          M=M+1
          TLIST(M,P)=TLIST(J,P)
        ENDIF
      ENDDO
      DO J=INDX(NUM)+1,INDL(P)
        M=M+1
        TLIST(M,P)=TLIST(J,P)
      ENDDO
      INDL(P)=M
    ENDIF
  C
  C   CHECK CONSISTENCY OF LISTS OF SURROUNDING PARTICLES.
  C-----
      NUM=0
      DO J=1,NNBSLIS
        K=NBSLIST(J)
        R=DSQRT((POS(P,1)-POS(K,1))**2+(POS(P,2)-POS(K,2))**2)
        IF(R.LT.H(K)/SOF.AND.R.GT.HP/SOF) THEN
          NUM=NUM+1
          WORKVEC(NUM)=R
          TEMPLIS(NUM)=K
        ENDIF
      ENDDO
      DO J=1,NACC
        K=ACCLIST(J)
        R=DSQRT((POS(P,1)-POS(K,1))**2+(POS(P,2)-POS(K,2))**2)
        IF(R.LT.H(K)/SOF.AND.R.GT.HP/SOF) THEN
          NUM=NUM+1
          WORKVEC(NUM)=R
          TEMPLIS(NUM)=K
        ENDIF
      ENDDO
      DO J=1,NUM
        R=WORKVEC(J)
        K=TEMPLIS(J)
        HIJ=(HP+H(K))*0.5D0/SOF
        IF(R.GT.HIJ) THEN
  C*VDIR: PREFER SCALAR
          DO L=1,NNB(K)
            IF(SLIST(L,K).EQ.P) THEN
              DO M=L,NNB(K)-1
                SLIST(M,K)=SLIST(M+1,K)

```

```

                                ENDDO
                                NNB(K)=NNB(K)-1
                                GOTO 60
                                ENDIF
                                ENDDO
60                                CONTINUE
                                ELSE
                                    DO L=1,INDL(P)
                                        IF(K.EQ.TLIST(L,P)) GOTO 40
                                    ENDDO
                                    TLIST(INDL(P)+1,P)=K
                                    INDL(P)=INDL(P)+1
40                                CONTINUE
                                ENDIF
                                ENDDO

                                ENDDO

CC35 CONTINUE
C
C  PUT PARTICLES IN TLIST INTO SLIST.
C-----

DO I=1,NNBSLIS
    P=NBSLIST(I)
    IF(NNB(P)+INDL(P).GT.3*NNBMAX) THEN
        WRITE(ULOG,*) P,NNB(P),INDL(P)
        CALL TERROR(' OVERFLOW IN INITLIS. ')
    ENDIF
ENDDO
DO I=1,NACC
    P=ACCLIST(I)
    IF(NNB(P)+INDL(P).GT.3*NNBMAX)
&    CALL TERROR(' OVERFLOW IN INITLIS  ')
ENDDO
DO I=1,NNBSLIS
    P=NBSLIST(I)
    NUM=NNB(P)
    DO J=1,INDL(P)
        NUM=NUM+1
        SLIST(NUM,P)=TLIST(J,P)
    ENDDO
    NNB(P)=NUM
ENDDO
DO I=1,NACC
    P=ACCLIST(I)
    NUM=NNB(P)
    DO J=1,INDL(P)

```

```

        NUM=NUM+1
        SLIST(NUM,P)=TLIST(J,P)
    ENDDO
    NNB(P)=NUM
ENDDO

C
C   BUILD LIST FROM SLIST.
C-----

DO I=1,NNBSLIS
    P=NBSLIST(I)
    NUM=0
    HP=H(P)
    DO J=1,NNB(P)
        K=SLIST(J,P)
        DR=DSQRT((POS(P,1)-POS(K,1))**2+(POS(P,2)-POS(K,2))**2)
        HIJ=(HP+H(K))*0.5D0
        IF(DR.LT.HIJ) THEN
            NUM=NUM+1
            INDX(NUM)=J
        ENDIF
    ENDDO
    IF(NUM.GT.2*NNBMAX) THEN
        WRITE(ULOG,*) P,NUM,POS(P,1),POS(P,2)
        CALL TERROR(' OVERFLOW IN LIST. ')
    ENDIF
    DO J=1,NUM
        LIST(J,P)=SLIST(INDX(J),P)
    ENDDO
    NNBS(P)=NUM
ENDDO

DO I=1,NACC
    P=ACCLIST(I)
    NUM=0
    HP=H(P)
    DO J=1,NNB(P)
        K=SLIST(J,P)
        HIJ=(HP+H(K))*0.5D0
        DR=DSQRT((POS(P,1)-POS(K,1))**2+(POS(P,2)-POS(K,2))**2)
        IF(DR.LT.HIJ) THEN
            NUM=NUM+1
            INDX(NUM)=J
        ENDIF
    ENDDO
    IF(NUM.GT.2*NNBMAX) THEN
        WRITE(ULOG,*) P,NUM,POS(P,1),POS(P,2)
        CALL TERROR(' OVERFLOW IN LIST ')
    ENDIF
ENDDO

```



```

PI=4.D0*DATAN(1.D0)
C1=40.D0/(7.D0*PI)
C2=240.D0/(7.D0*PI)
IF(NMP.NE.0) MASS1=MASS(NMP)
ARAD=H(1)

```

```

C=====

```

```

C  INITIALIZE MISC. USEFUL NUMBERS.

```

```

C  -----

```

```

      IDUM=-1
      MINUST = -2.D0
      TINY=1.E-20
      ZERO=0.D0
      ONE=1.D0
      TWO=2.D0
      GMONE=GAMMA-ONE
      GMTWO=GAMMA-TWO
      MDOT=0.D0
      TAM=0.D0
      DO K=1,3
        TAAM(K)=0.D0
      ENDDO
      TAEK=0.D0
      TAE=0.D0
      NAGP=0
      NACC=0
      BNOW=0
      SOF=0.8D0
      DO I=0,50
        LT(I)=0.D0
      ENDDO
      RSIZE=0.
      DO I=1,NDIM
        RBMIN(I)=0.
      ENDDO

```

```

C  INITIALIZE SIZE PARAMETER FOR BODIES.

```

```

C  -----

```

```

      DO 5 P=1,NBODIES
        CELLSIZ(P)=0.
        BODLIST(P)=P
        ADVLIST(P)=P
        TBIN(P)=-1
5      CONTINUE
      NBODLIS=NBODIES
      NADVLIS=NBODIES
      NGAS=0

```

```

I=0
DOWHILE (ADVLIST(I+1) .LE. NGP .AND. (I+1) .LE. NADVLIST)
  I=I+1
ENDDO
NGAS=I
DO K=1,NDIM
  DO P=1,NBODIES
    DVEL(P,K)=VEL(P,K)
  ENDDO
ENDDO
DO I=1+NMP,NGP
  NBSLIST(I-NMP)=BODLIST(I)
ENDDO
NNBSLIST=NGP-NMP

```

C INITIALIZE CENTER OF MASS DATA.

C-----

```

DO K=1,NDIM
  CMVEL(K)=0.D0
  CMPOS(K)=0.D0
ENDDO
MTOT=0.D0
DO I=1,NBODLIST
  P=BODLIST(I)
  MTOT=MTOT+MASS(P)
ENDDO

DO K=1,NDIM
  DO I=1,NBODLIST
    P=BODLIST(I)
    CMVEL(K)=CMVEL(K)+MASS(P)*VEL(P,K)
    CMPOS(K)=CMPOS(K)+MASS(P)*POS(P,K)
  ENDDO
  CMPOS(K)=CMPOS(K)/MTOT
  CMVEL(K)=CMVEL(K)/MTOT
ENDDO
DO K=1,NDIM
  POS(0,K)=CMPOS(K)
ENDDO

```

C RESET 'GRID' PARTICLES FOR SPECTRAL TIME ANALYSIS.

C-----

```

DELR=(1.D0-0.1D0)/DFLOAT(NGPTS-1)
DO I=NBODIES+1,NBODIES+NGPTS
  POS(I,1)=0.1D0+DELR*(I-1-NBODIES)+CMPOS(1)
  POS(I,2)=CMPOS(2)
ENDDO
DO I=NBODIES+1+NGPTS,NBODIES+2*NGPTS

```

```

      POS(I,1)=-0.1D0-DELR*(I-1-NBODIES-NGPTS)+CMPOS(1)
      POS(I,2)=CMPOS(2)
    ENDDO
    DO I=NBODIES+1+2*NGPTS,NBODIES+3*NGPTS
      POS(I,2)=0.1D0+DELR*(I-1-NBODIES-2*NGPTS)+CMPOS(2)
      POS(I,1)=CMPOS(1)
    ENDDO
    DO I=NBODIES+1+3*NGPTS,NBODIES+4*NGPTS
      POS(I,2)=-0.1D0-DELR*(I-1-NBODIES-3*NGPTS)+CMPOS(2)
      POS(I,1)=CMPOS(1)
    ENDDO

C
C  INITIALIZE GAS VELOCITIES
C-----
      DO K=1,NDIM
        DO I=1,NBODIES
          VET(I,K)=VEL(I,K)
        ENDDO
      ENDDO

C  INITIALIZE TIME PARAMETERS.
C-----
      IF(TNOW.EQ.0.) THEN
        TDUMP=TD
      ELSE
        TDUMP=DINT(TNOW/TD)*TD + TD
      ENDIF
      TPOS=TNOW
      DTS=0.D0
      TOL2INV=1./(TOL*TOL)

C-----
C  INITIALIZE VARIABLES AND ARRAYS FOR GRAVITATIONAL FIELD SMOOTHING
C  INTERPOLATION.  INTERPOLATION PERFORMED IN DISTANCE.
C-----
      DELDRG=2./NINTERP

      DO 30 I=0,1+NINTERP
        XW=I*DELDRG
        XW2=XW*XW
        XW3=XW2*XW
        XW4=XW2*XW2
        PHSMOOT(I)=CVMGT(-2.D0*XW3*(ONE/3.-3.*XW2/20.+XW3/20.)+
&                      7.*XW/5.,-ONE/15.+8.*XW/5.-XW3*(4./3.-XW+
&                      3.*XW2/10.-XW3/30.),XW.LE.ONE)
        PHSMOOT(I)=CVMGT(ONE,PHSMOOT(I),XW.GE.TWO)
        ACSMOOT(I)=CVMGT(XW3*(4./3.-6.*XW2/5.+0.5*XW3),-ONE/15.+

```

```

&          8.*XW3/3.-3.*XW4+6.*XW3*XW2/5.-XW4*XW2/6.,
&          XW.LE.ONE)
      ACSMOOT(I)=CVMGT(ONE,ACSMOOT(I),XW.GE.TWO)
30      CONTINUE

      RETURN
      END
@PROCESS DC(BODYCEL,CONCOM,GAS)
@PROCESS DIRECTIVE('*VDIR:')
C*****
C
C
C          SUBROUTINE INITPOS
C
C
C*****
C
C
C      SUBROUTINE TO APPLY CORRECTION FACTOR TO MAKE THE LEAP-FROG
C      ALGORITHM SELF-STARTING (HERNQUIST AND KATZ, AP. J. SUPPL.,
C      IN PRESS [1988]).
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C  DECLARATION OF LOCAL VARIABLES.
C  -----

      DOUBLE PRECISION DT2,R
      INTEGER P,K,I

C=====

C  LOOP OVER ALL SPATIAL COORDINATES FOR ALL BODIES.
C  -----

      DT2=DTIME**2

      DO 200 K=1,NDIM
C*VDIR: IGNORE RECRDEPS
          DO 100 I=1,NBODLIS
              P=BODLIST(I)
              POS(P,K)=POS(P,K)+ACC(P,K)*DT2*0.125D0
100          CONTINUE
200          CONTINUE

```

```

C      MOVE ACCRETED PARTICLES WITH P=1
C-----
C      DO K=1,NDIM
C*VDIR: IGNORE RECRDEPS
C      DO I=1,NACC
C      P=ACCLIST(I)
C      POS(P,K)=POS(P,K)+ACC(1,K)*DT2*0.125D0
C      ENDDO
C      ENDDO

C      MTOT=0.D0
C      DO I=1,NBODLIS
C      P=BODLIST(I)
C      MTOT=MTOT+MASS(P)
C      ENDDO
C      CMPOS(1)=0.D0
C      CMPOS(2)=0.D0
C      DO K=1,NDIM
C      DO I=1,NBODLIS
C      P=BODLIST(I)
C      CMPOS(K)=CMPOS(K)+MASS(P)*POS(P,K)
C      ENDDO
C      CMPOS(K)=CMPOS(K)/MTOT
C      ENDDO
C      DO I=1,NBODIES
C      R=DSQRT((POS(I,1)-CMPOS(1))**2+(POS(I,2)-CMPOS(2))**2)
C      MINDX(I)=INT(R/ANW)+1
C      ENDDO

C      RETURN
C      END

@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
C
C      SUBROUTINE INIT'SOF
C
C
C*****
C
C
C      SUBROUTINE TO SOFTEN INITIAL CONDITIONS, AND TO FIND THE
C      INITIAL LIST OF NEIGHBORS AND INITIAL DENSITY.
C      R. DRIMMEL 4/92
C
C=====

```

```
/*INCLUDE NEWHEAD FORTRAN A
```

```
C  DECLARATION OF LOCAL VARIABLES.
```

```
C  -----
```

```
      INTEGER I,J,K,P,FLAG,IOLD,NVAL,INDX(NSTMAX),I1,L
      DOUBLE PRECISION DR,UH,AA,DL,DX,HINV,H2INV,R,
&      DY,FPI,NBDR(NSTMAX),MASSP
```

```
      CALL MAKETRE
```

```
C  -----
```

```
      IF(TNOW.EQ.0.D0) THEN
```

```
C-----
```

```
C
```

```
C  ADJUST H(I)
```

```
C-----
```

```
      IOLD=0
```

```
      DO P=NMP+1,NGP
```

```
        FLAG=0
```

```
65      FLAG=FLAG+1
```

```
        IF(FLAG.GE.30) THEN
```

```
          WRITE(ULOG,*) P,POS(P,1),POS(P,2),H(P)
```

```
          CALL ERROR(' UNABLE TO ADJUST SMOOTHING ')
```

```
        ENDIF
```

```
        CALL WALKIT(P,H(P),NVAL)
```

```
        IF(NVAL.LT.(8*NNBMAX)/10) THEN
```

```
          H(P)=H(P)*1.1D0
```

```
          GOTO 65
```

```
        ENDIF
```

```
        IF(NVAL.GT.NSTMAX) THEN
```

```
          WRITE(ULOG,*) '          NVAL.GT.200 FOR PART.',P
```

```
          WRITE(ULOG,*) '          H=',H(P), 'TIME=',TNOW
```

```
          H(P)=H(P)*0.5D0
```

```
          GOTO 65
```

```
        ENDIF
```

```
        DO J=1,NVAL
```

```
          NBDR(J)=WORKVEC(J)
```

```
        ENDDO
```

```
        IF(NVAL.GT.NNBMAX) THEN
```

```
          CALL PKSRT(NVAL,NBDR,INDX)
```

```
          H(P)=(NBDR(INDX(NNBMAX))+NBDR(INDX(8*NNBMAX/10)))*0.5D0
```

```
          CALL WALKIT(P,H(P),NVAL)
```

```
          IF(NVAL.GT.NNBMAX.OR.NVAL.LT.(8*NNBMAX/10)) THEN
```

```
            CALL ERROR(' ERROR IN INITSOE. ')
```

```
          ENDIF
```

```
        ENDIF
```

```
      ENDDO
```

```
    ENDIF
```

```

C
C   SOFTEN INITIAL CONDITIONS
C-----
C
C   DO I=NMP+1,NGP
C       FPI=MASS(I)*C1/(H(I)*H(I)*RH(I))
C       ACC(I,1)=VEL(I,1)*FPI
C       ACC(I,2)=VEL(I,2)*FPI
C       DO L=1,NNBS(I)
C           J=LIST(L,I)
C           HINV=2.D0/(H(I)+H(J))
C           H2INV=HINV*HINV
C           DX=POS(I,1)-POS(J,1)
C           DY=POS(I,2)-POS(J,2)
C           DR=DSQRT(DX*DX+DY*DY)
C           UH=DR*HINV
C           AA=NINTERP*UH
C           I1=AA
C           DL=AA-I1
C           DR=1.-DL
C           I1=I1+1
C           FPI=(DR*VW3(I1)+DL*VW3(I1+1))*C1*MASS(J)*H2INV/RH(J)
C           ACC(I,1)=ACC(I,1)+VEL(J,1)*FPI
C           ACC(I,2)=ACC(I,2)+VEL(J,2)*FPI
C       ENDDO
C   ENDDO
C   DO K=1,NDIM
C       DO I=NMP+1,NGP
C           VEL(I,K)=ACC(I,K)
C       ENDDO
C   ENDDO

C
C   WRITE OUT NEW SOFTENED INITIAL CONDITIONS
C-----
C       WRITE(20,100) NBODIES,NDIM,TNOW,NGP,NMP,QTOT
C100   FORMAT(1X,A,3X,2(I5,3X),G21.14,3X,2(I5,3X))

C       MASSP=MASS(NMP)
C       MASS(NMP)=MASS1
C       DO 10 I=1,NGP
C           WRITE(20,*) POS(I,1),POS(I,2)
C           WRITE(20,*) VEL(I,1),VEL(I,2)
C           WRITE(20,*) MASS(I),H(I)
C10    CONTINUE
C       MASS(NMP)=MASSP

C       DO 20 I=NGP+1,NBODIES

```

```

C          WRITE(20,*) POS(I,1),POS(I,2)
C          WRITE(20,*) VEL(I,1),VEL(I,2)
C          WRITE(20,*) MASS(I)
C20      CONTINUE
C
C      ENDIF

```

```

      RETURN
      END

```

```

@PROCESS DC(BODYCEL,CONCOM,GAS)

```

```

C*****

```

```

C

```

```

C

```

```

          SUBROUTINE INITSYS(N)

```

```

C

```

```

C

```

```

C*****

```

```

C

```

```

C

```

```

C      SUBROUTINE TO INITIALIZE THE STATE OF THE SYSTEM.

```

```

C

```

```

C

```

```

C=====

```

```

/*INCLUDE NEWHEAD FORTRAN A

```

```

C=====

```

```

      INTEGER N

```

```

C      BEGIN TIMING.

```

```

C      -----

```

```

          CALL SECONDS(CPUT0)

```

```

C-----

```

```

C      OPEN DATA FILES, READ INPUT PARAMETERS AND INITIAL SYSTEM STATE,
C      AND INITIALIZE SYSTEM PARAMETERS.

```

```

C-----

```

```

          CALL STARTOU

```

```

C      -----

```

```

          CALL INPARAM

```

```

C      -----

```

```

          CALL INBODS

```

```

C      -----

```

```

          CALL INITPAR

```

```

C      -----

```

```

          CALL INITSO

```

```

C          -----
C
C  COMPUTE GRAVITATIONAL POTENTIAL AND ACCELERATION.
C  -----
C      CALL ZEROACC
C      -----
C      CALL ZEROPOT
C      -----
C
C  CALCULATE THE INITIAL DENSITY
C  -----

      IF (NMP.NE.0.OR.TNOW.EQ.0.DO) THEN
        CALL INITLIS
C      -----
C      CALL DENSITY
C      -----
      ENDIF

C
C  CALCULATE INITIAL INTERNAL ENERGIES (FOR POLYTROPIC EQ. OF STATE).
C  -----
      IF (TNOW.EQ.0.DO) THEN
        DO I=NMP+1,NGP
          U(I)=ISS*RH(I)**GMONE/GMONE
        ENDDO
      ENDIF

      CALL ACCRET(0)
C      -----
C
C  REINITIALIZE ADVLIST AND NBSLIST.
C  -----
      IF (NMP.NE.0) THEN
        DO I=1,NBODLIS
          ADVLIST(I)=BODLIST(I)
        ENDDO
        NADVLIS=NBODLIS
        NGAS=0
        I=0
        DOWHILE (ADVLIST(I+1).LE.NGP.AND.(I+1).LE.NADVLIS)
          I=I+1
        ENDDO
        NGAS=I
        DO I=1+NMP,NGP-NAGP
          NBSLIST(I-NMP)=BODLIST(I)
        ENDDO
        NNBSLIS=NGP-NMP-NAGP

```

```

ENDIF

CALL GRAVITY('BOTH')
C -----
CALL INITLIS
C -----
CALL DENSITY
C -----
CALL HYDRO(0)
C -----
CALL COURANT(N)
C -----
CALL AMFLUX
C -----
CALL GRIDEN
C -----

C OUTPUT SYSTEM STATE.
C -----
CALL OUTSTAT(0)
C -----
C
C THE FIRST STEP.
C -----
DO K=1,NDIM
  DO I=1,NADVLIS
    P=ADVLIST(I)
    VEL(P,K)=VEL(P,K)+0.25D0*ACC(P,K)*DTS/(2.D0**TBIN(P))
  ENDDO
ENDDO
NCMAX=NMAX

CALL ESTVEL
C -----
CALL STEPPOS
C -----
C
C MAKE LIST OF PARTICLES TO BE ADVANCED IN THE NEXT HALF STEP.
C-----
BNOW=NMAX
CALL WHENIEQ(NBODIES,TBIN,1,BNOW,ADVLIST,NADVLIS)
C -----
C WRITE(22,*) NMAX,BNOW,NADVLIS,NNBSLIS
NGAS=0
I=0
DOWHILE(ADVLIST(I+1).LE.NGP.AND.(I+1).LE.NADVLIS)
  I=I+1
ENDDO

```

```

      NGAS=I

      CALL HALFSTP(N)
C      -----
      CALL AMFLUX
C      -----
      CALL ACCRET(N)
C      -----
      CALL COURANT(N)
C      -----

      RETURN
      END
@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
C      SUBROUTINE INPARAM
C
C
C*****
C
C
C      SUBROUTINE TO READ IN PARAMETERS.
C
C      INPUT PARAMETERS:
C
C      HEADLIN      : IDENTIFICATION STRING FOR THE RUN.
C      NSTEPS       : NUMBER OF TIMESTEPS.
C      TD           : OUTPUT SYSTEM STATE ONCE EVERY TIME TD.
C      NOUTLOG      : OUTPUT LOGFILE DATA ONCE EVERY NSTEPS/NOUTLOG
C                   STEPS.
C      CN           : THE COURANT NUMBER
C      ZETA         : NEIGHBOR FINDING PARAMETER
C      ANW          : WIDTH OF ANNULI FOR VELOCITY AND AM ANALYSIS.
C      TOL          : ERROR TOLERANCE; 0.0 => EXACT (PP) CALCULATION.
C      EPS          : POTENTIAL SOFTENING PARAMETER.
C      VTYPE        : VISCOCITY TYPE : 1=MONAGHAN, 2=HERNQUIST
C      ALFA,BHTA    : BULK VISCOSITY PARAMETERS
C      NU           : SHEAR VISCOSITY PARAMETER
C      SWITCH       : VISCOSITY SWITCH
C      USEQUAD      : OPTION TO INCLUDE (.TRUE.) QUADRUPOLE TERMS.
C      ISS          : POLYTROPIC GAS CONSTANT (ISOTHERMAL SOUND SPEED)
C      GAMMA        : INDEX IN EQUATION OF STATE
C
C=====

```

```

/*INCLUDE NEWHEAD FORTRAN A

```

```

=====
C   READ PARAMETERS, CLOSE THE FILE.
C   -----
C       READ(UPARS, '(A)') HEADLIN
C       READ(UPARS, *) NSTEPS
C       READ(UPARS, *) TD
C       READ(UPARS, *) NOUTLOG
C       READ(UPARS, *) CN
C       READ(UPARS, *) ZETA, ANW
C       READ(UPARS, *) EPS, TOL
C       READ(UPARS, *) VTYP, ALFA, BHTA, NU
C       READ(UPARS, *) SWITCH
C       READ(UPARS, *) USEQUAD
C       READ(UPARS, *) ISS, GAMMA
C       CLOSE(UNIT=UPARS)
C
C       RETURN
C       END
C@PROCESS DC(BODYCEL, CONCOM, GAS)
C@PROCESS DIRECTIVE(' *VDIR: ')
C*****
C
C
C       SUBROUTINE LOADSAP
C
C
C*****
C
C
C       SUBROUTINE TO INSERT THE BODIES INTO THE SAPLING. THE PROCESS
C       IS VECTORIZED OVER ACTIVE BODIES (MAKINO, J. COMPUT. PHYS.,
C       SUBMITTED [1988]). ACTIVE BODIES ARE THOSE WHICH ARE NOT YET IN
C       PLACE IN THE TREE, AS LEAVES.  ADAPTED FROM SUBROUTINE LOADTRE.
C       R. DRIMMEL (APRIL 16, 1994)
C
C
C=====
C/*INCLUDE NEWHEAD FORTRAN A
C
C   DECLARATION OF LOCAL VARIABLES.
C   -----
C
C       DOUBLE PRECISION PM1(NSUBCEL, NDIM)
C       INTEGER K, P, NINDEX(NDIM), J, I, NACTLIS, NCLIST, NCLIST2,
C       & NSUBSET, INDCELL, NSUBBOD, NBODTEM

```

```

      SAVE NINDEX, PM1

      DATA PM1/2*-1., 2*1., -1., +1., -1., +1./, NINDEX/2, 1/

C=====

C  DEALLOCATE OLD TREE, COMPUTE COORDINATES OF CENTER OF ROOT CELL.
C  -----
      INCELLS=1
      ROOT=NBODSMA+1

      DO 5 J=1, NSUBCEL
        SUBP(ROOT, J)=0
5      CONTINUE

      CELLSIZ(ROOT)=RSIZE

      DO 10 K=1, NDIM
        POS(ROOT, K)=RBMIN(K)+0.5D0*RSIZE
10     CONTINUE

C-----
C  PLACE ALL BODIES ON ACTIVE BODY LIST, HAVING ROOT AS PARENT; PLACE
C  ROOT ON ACTIVE CELL LIST.
C-----

      DO 20 I=1, NNBSLIS
        PARENT(I)=ROOT
        ACTLIST(I)=NBSLIST(I)
20     CONTINUE

      NACTLIS=NNBSLIS
      CELLIST(1)=ROOT
      NCLIST=1

C  LOOP UNTIL NO BODIES ARE LEFT ACTIVE.
C  -----

200    CONTINUE

      IF(NCLIST.GT.0) THEN

C  COMPUTE SUBINDICES FOR ALL ACTIVE BODIES.
C  -----
      DO 30 I=1, NACTLIS
        SUBINDE(I)=1
30     CONTINUE

```

```

DO 50 K=1,NDIM
  DO 40 I=1,NACTLIS
    IF(POS(ACTLIST(I),K).GE.POS(PARENT(I),K))
      &      SUBINDE(I)=SUBINDE(I)+NINDEX(K)
40      CONTINUE
50      CONTINUE

C  COMPUTE NUMBER OF BODIES IN EACH SUBCELL.
C  -----
C*VDIR: IGNORE RECRDEPS
      DO 60 I=1,NACTLIS
        SUBP(PARENT(I),SUBINDE(I))=SUBP(PARENT(I),SUBINDE(I))+1
60      CONTINUE

C-----
C  OPEN ALL SUBCELLS WITH MORE THAN ONE BODY, PLACING THEM ON ACTIVE
C  CELL LIST.
C-----
      NCLIST2=0

      DO 110 J=1,NSUBCEL

        DO 70 I=1,NCLIST
          ASUBP(I)=SUBP(CELLIST(I),J)
70      CONTINUE

        CALL WHENIGT(NCLIST,ASUBP,1,1,ISUBSET,NSUBSET)

        INCELLS=INCELLS+NSUBSET
        IF(INCELLS.GT.NCELLS) THEN
          &      WRITE(ULOG,*) ' INCELLS,NSUBSET,NACTLIS=',INCELLS,
          &      NSUBSET,NACTLIS,NNBSLIS,NADVLIS,BNOW,NMAX
          DO I=1,NACTLIS
            WRITE(ULOG,*) ACTLIST(I),TBIN(ACTLIST(I))
          ENDDO
          CALL ERROR(' OVERFLOW IN LOADSAP')
C      -----
        ENDIF
        INDCELL=INCELLS-NSUBSET+NBODSMA

        DO 90 K=1,NSUBCEL
          DO 80 I=1,NSUBSET
            SUBP(INDCELL+I,K)=0
80          CONTINUE
90          CONTINUE

C*VDIR: IGNORE RECRDEPS
      DO 100 I=1,NSUBSET

```

```

      P=INDCELL+I
      ASUBP(I)=CELLIST(ISUBSET(I))
      SUBP(ASUBP(I),J)=P
      CELLSIZ(P)=CELLSIZ(ASUBP(I))*0.5
      TEMPLIS(NCLIST2+I)=P
      POS(P,1)=POS(ASUBP(I),1)+PM1(J,1)*0.5D0*CELLSIZ(P)
      POS(P,2)=POS(ASUBP(I),2)+PM1(J,2)*0.5D0*CELLSIZ(P)
      POS(P,3)=POS(ASUBP(I),3)+PM1(J,3)*0.5D0*CELLSIZ(P)
C
100      CONTINUE

      NCLIST2=NCLIST2+NSUBSET

110      CONTINUE

      NCLIST=NCLIST2

      DO 120 I=1,NCLIST
        CELLIST(I)=TEMPLIS(I)
120      CONTINUE

C  FIND ALL SUBCELLS WITH ONE BODY; ADD BODIES TO TREE.
C  -----
      DO 130 I=1,NACTLIS
        TEMPLIS(I)=NCELLS*(SUBINDE(I)-1)+(PARENT(I)-NBODSMA)
        ASUBP(I)=SUBPVEC(TEMPLIS(I))
130      CONTINUE

      CALL WHENEQ(NACTLIS,ASUBP,1,1,ISUBSET,NSUBBOD)

      DO 140 I=1,NSUBBOD
        SUBPVEC(TEMPLIS(ISUBSET(I)))=ACTLIST(ISUBSET(I))
140      CONTINUE

C  PLACE BODIES IN CELLS WITH MORE THAN ONE BODY ON ACTIVE LIST.
C  -----

      CALL WHENIGT(NACTLIS,ASUBP,1,1,ISUBSET,NBODTEM)

      NACTLIS=NBODTEM

      DO 150 I=1,NACTLIS
        PARENT(I)=ASUBP(ISUBSET(I))
        TEMPLIS(I)=ACTLIST(ISUBSET(I))
150      CONTINUE

      DO 160 I=1,NACTLIS
        ACTLIST(I)=TEMPLIS(I)
160      CONTINUE

```

```

      GO TO 200

      ENDIF

      RETURN
      END@PROCESS DC(BODYCEL,CONCOM,GAS)
@PROCESS DIRECTIVE('*VDIR:')
C*****
C
C
C          SUBROUTINE LOADTRE
C
C
C*****
C
C
C      SUBROUTINE TO INSERT THE BODIES INTO THE TREE.  THE PROCESS IS
C      VECTORIZED OVER ACTIVE BODIES (MAKINO, J. COMPUT. PHYS.,
C      SUBMITTED [1988]).  ACTIVE BODIES ARE THOSE WHICH ARE NOT YET IN
C      PLACE IN THE TREE, AS LEAVES.  THE LOCAL VARIABLES PM1 AND NINDEX
C      ARE USED TO CONVERT BACK AND FORTH BETWEEN PHYSICAL COORDINATES
C      AND SUBCELL COORDINATES.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C  DECLARATION OF LOCAL VARIABLES.
C  -----

      DOUBLE PRECISION PM1(NSUBCEL,NDIM)
      INTEGER K,P,NINDEX(NDIM),J,I,NACTLIS,NCLIST,NCLIST2,
&          NSUBSET,INDCELL,NSUBBOD,NBODTEM

      SAVE NINDEX,PM1

      DATA PM1/2*-1.,2*1.,-1.,+1.,-1.,+1./,NINDEX/2,1/

C=====

C  DEALLOCATE OLD TREE, COMPUTE COORDINATES OF CENTER OF ROOT CELL.
C  -----
      INCELLS=1
      ROOT=NBODSMA+1

      DO 5 J=1,NSUBCEL

```

```

      SUBP(ROOT,J)=0
5      CONTINUE

      CELLSIZ(ROOT)=RSIZE

      DO 10 K=1,NDIM
        POS(ROOT,K)=RBMIN(K)+0.5D0*RSIZE
10     CONTINUE

C-----
C   PLACE ALL BODIES ON ACTIVE BODY LIST, HAVING ROOT AS PARENT; PLACE
C   ROOT ON ACTIVE CELL LIST.
C-----

      DO 20 I=1,NBODLIS-NMP
        PARENT(I)=ROOT
        ACTLIST(I)=BODLIST(I+NMP)
20     CONTINUE

      NACTLIS=NBODLIS-NMP
      CELLIST(1)=ROOT
      NCLIST=1

C   LOOP UNTIL NO BODIES ARE LEFT ACTIVE.
C   -----

200    CONTINUE

      IF(NCLIST.GT.0) THEN

C   COMPUTE SUBINDICES FOR ALL ACTIVE BODIES.
C   -----
        DO 30 I=1,NACTLIS
          SUBINDE(I)=1
30     CONTINUE

        DO 50 K=1,NDIM
          DO 40 I=1,NACTLIS
            IF(POS(ACTLIST(I),K).GE.POS(PARENT(I),K))
              SUBINDE(I)=SUBINDE(I)+NINDEX(K)
          &
40     CONTINUE
50     CONTINUE

C   COMPUTE NUMBER OF BODIES IN EACH SUBCELL.
C   -----
C*VDIR: IGNORE RECRDEPS
        DO 60 I=1,NACTLIS
          SUBP(PARENT(I),SUBINDE(I))=SUBP(PARENT(I),SUBINDE(I))+1

```

```

60          CONTINUE

C-----
C  OPEN ALL SUBCELLS WITH MORE THAN ONE BODY, PLACING THEM ON ACTIVE
C  CELL LIST.
C-----

          NCLIST2=0

          DO 110 J=1, NSUBCEL

              DO 70 I=1, NCLIST
                  ASUBP(I)=SUBP(CELLIST(I), J)
70          CONTINUE

              CALL WHENIGT(NCLIST, ASUBP, 1, 1, ISUBSET, NSUBSET)

              INCELLS=INCELLS+NSUBSET
              IF (INCELLS.GT.NCELLS) THEN
                  WRITE(ULOG,*) ' INCELLS, NSUBSET', INCELLS, NSUBSET
                  CALL TERROR(' OVERFLOW IN LOADTRE')
C          -----
              ENDIF
              INDCELL=INCELLS-NSUBSET+NBODSMA

              DO 90 K=1, NSUBCEL
                  DO 80 I=1, NSUBSET
                      SUBP(INDCELL+I, K)=0
80          CONTINUE
90          CONTINUE

C*VDIR: IGNORE RECRDEPS
          DO 100 I=1, NSUBSET
              P=INDCELL+I
              ASUBP(I)=CELLIST(ISUBSET(I))
              SUBP(ASUBP(I), J)=P
              CELLSIZ(P)=CELLSIZ(ASUBP(I))*0.5
              TEMPLIS(NCLIST2+I)=P
              POS(P, 1)=POS(ASUBP(I), 1)+PM1(J, 1)*0.5D0*CELLSIZ(P)
              POS(P, 2)=POS(ASUBP(I), 2)+PM1(J, 2)*0.5D0*CELLSIZ(P)
C          POS(P, 3)=POS(ASUBP(I), 3)+PM1(J, 3)*0.5D0*CELLSIZ(P)
100         CONTINUE

          NCLIST2=NCLIST2+NSUBSET

110        CONTINUE

          NCLIST=NCLIST2

```

```

DO 120 I=1,NCLIST
  CELLIST(I)=TEMPLIS(I)
120  CONTINUE

C  FIND ALL SUBCELLS WITH ONE BODY; ADD BODIES TO TREE.
C  -----
      DO 130 I=1,NACTLIS
        TEMPLIS(I)=NCELLS*(SUBINDE(I)-1)+(PARENT(I)-NBODSMA)
        ASUBP(I)=SUBPVEC(TEMPLIS(I))
130  CONTINUE

      CALL WHENEQ(NACTLIS,ASUBP,1,1,ISUBSET,NSUBBOD)

      DO 140 I=1,NSUBBOD
        SUBPVEC(TEMPLIS(ISUBSET(I)))=ACTLIST(ISUBSET(I))
140  CONTINUE

C  PLACE BODIES IN CELLS WITH MORE THAN ONE BODY ON ACTIVE LIST.
C  -----

      CALL WHENIGT(NACTLIS,ASUBP,1,1,ISUBSET,NBODTEM)

      NACTLIS=NBODTEM

      DO 150 I=1,NACTLIS
        PARENT(I)=ASUBP(ISUBSET(I))
        TEMPLIS(I)=ACTLIST(ISUBSET(I))
150  CONTINUE

      DO 160 I=1,NACTLIS
        ACTLIST(I)=TEMPLIS(I)
160  CONTINUE

      GO TO 200

ENDIF

RETURN
END
@PROCESS DC(BODYCEL,CONCOM,GAS)
@PROCESS DIRECTIVE(' *VDIR:')
C-----
C
C
      SUBROUTINE MAKELIS
C
C
C-----

```

```

C
C
C      SUBROUTINE TO BUILD LIST OF NEIGHBORS FOR ALL PARTICLES.
C
C
C=====
/*INCLUDE NEWHEAD FORTRAN A

      INTEGER NVAL, I, J, K, L, P, M, NUM, FLAG, INDX(1000), NUMAX, PMAX, NBS,
&      TLIST(2*NNBMAX, NGPMAX), COUNT2, COUNT1, CLIST(20*NNBMAX),
&      NADJ, NFIX, NBSMIN, FLAG2
      DOUBLE PRECISION DR, HIJ, HP, HOLD, RAD(1000), R, UH, AVADJ

C
C
C      FIND NEAREST NEIGHBORS OF ACTIVE PARTICLES
C-----
      DO I=NMP+1, NGP-NAGP
        P=BODLIST(I)
        INDL(P)=0
        NNB(P)=0
      ENDDO
      DO I=1, NACC
        P=ACCLIST(I)
        INDL(P)=0
        NNB(P)=0
      ENDDO
      DO I=1, 20*NNBMAX
        CLIST(I)=0
      ENDDO
      NUMAX=0
      COUNT1=0
      COUNT2=0
      NFIX=0
      AVADJ=0.D0
      DO I=1, NNBSLIS
        P=NBSLIST(I)
        NBSMIN=10
        FLAG=0
        FLAG2=0
        CALL WALKIT(P, (H(P)/SOF), NVAL)
15      NUM=0
        NBS=0
        DO J=1, NVAL
          K=TEMPLIS(J)
          HIJ=(H(K)+H(P))*0.5D0
          IF(WORKVEC(J).LE.HIJ/SOF) THEN
            NUM=NUM+1

```

```

        INDX(NUM)=J
        IF (WORKVEC(J).LE.HIJ) THEN
            NBS=NBS+1
        ENDIF
    ENDIF
ENDDO
IF (TBIN(P).EQ.BNOW) THEN
IF (NBS.GT.3*NNBMAX/2.AND.FLAG2.NE.1) THEN
    FLAG2=1
    COUNT2=COUNT2+1
    CLIST(COUNT2+18*NNBMAX)=P
    H(P)=0.8D0*H(P)
    GOTO 15
ENDIF
IF (NUM.GT.NUMAX) THEN
    PMAX=P
    NUMAX=NUM
    HP=H(P)
    L=NVAL
ENDIF
IF (NUM.GT.5*NNBMAX/2.OR.NBS.LT.NBSMIN) THEN
    IF (FLAG.EQ.0) THEN
        FLAG=1
        HOLD=H(P)
        CALL HPFIX(P,NVAL,NUM,NBS,NADJ)
        AVADJ=AVADJ+NADJ
        NFIX=NFIX+1
        IF (H(P).GT.HOLD) THEN
            COUNT1=COUNT1+1
            CLIST(COUNT1)=P
        ELSE
            COUNT2=COUNT2+1
            CLIST(18*NNBMAX+COUNT2)=P
        ENDIF
        IF (NBS.LT.NBSMIN) NBSMIN=NBS
        GOTO 15
    ELSE
        WRITE(ULOG,*) '    NUM=',NUM,'    NVAL=',NVAL,'    NBS=',NBS
        CALL TERROR(' ERROR IN MAKELIS. ')
    ENDIF
ENDIF
NNB(P)=NUM
DO J=1,NUM
    SLIST(J,P)=TEMPLIS(INDX(J))
ENDDO
ELSEIF (NUM.GT.1000) THEN
    WRITE(ULOG,400) P,NUM,TBIN(P),BNOW,POS(P,1),POS(P,2),H(P)
    FORMAT(1X,4(I5,1X),3(1PE12.5,2X))

```

```

      CALL TERROR(' OVERFLOW IN INDX IN MAKELIS. ')
    ENDIF
    DO J=1,NUM
      RAD(J)=WORKVEC(INDX(J))
    ENDDO
    DO J=1,NUM
      K=TEMPLIS(INDX(J))
      IF(RAD(J).GT.(H(K)/SOF)) THEN
        INDL(K)=INDL(K)+1
        TLIST(INDL(K),K)=P
      ENDIF
    ENDDO
  ENDDO
C
C   FIND NEAREST NEIGHBORS TO ACCRETED PARTICLES
C
C-----
    IF(BNOW.GE.TBIN(NMP)-2) THEN
      DO I=1,NACC
        P=ACCLIST(I)
        FLAG=0
        NBSMIN=10
        CALL WALKIT(P,(H(P)/SOF),NVAL)
25      NUM=0
        NBS=0
        DO J=1,NVAL
          HIJ=(H(P)+H(TEMPLIS(J)))*0.5D0
          IF(WORKVEC(J).LT.HIJ/SOF) THEN
            NUM=NUM+1
            INDX(NUM)=J
            IF(WORKVEC(J).LT.HIJ) THEN
              NBS=NBS+1
            ENDIF
          ENDIF
        ENDDO
        IF(NUM.GT.NUMAX) THEN
          PMAX=P
          NUMAX=NUM
          HP=H(P)
          L=NVAL
        ENDIF
        IF(NUM.GT.5*NNBMAX/2) THEN
          IF(FLAG.EQ.0) THEN
            FLAG=1
            HOLD=H(P)
            CALL HPFIX(P,NVAL,NUM,NBS,NADJ)
            AVADJ=AVADJ+NADJ
            NFIX=NFIX+1
          
```

```

      IF (H(P).GT.HOLD) THEN
        COUNT1=COUNT1+1
        CLIST(COUNT1)=P
      ELSE
        COUNT2=COUNT2+1
        CLIST(18*NNBMAX+COUNT2)=P
      ENDIF
      IF (NBS.LT.NBSMIN) NBSMIN=NBS
      GOTO 25
    ELSE
      WRITE(ULOG,*) '   NUM=',NUM,'   NVAL=',NVAL,'   NBS=',NBS
      CALL 'ERROR(' ERROR IN MAKELIS ')
    ENDIF
  ENDIF
ENDIF
DO J=1,NUM
  K=TEMPLIS(INDX(J))
  IF (WORKVEC(INDX(J)).GT.(H(K)/SOF)) THEN
    INDL(K)=INDL(K)+1
    TLIST(INDL(K),K)=P
  ENDIF
ENDDO
DO J=1,NUM
  K=TEMPLIS(INDX(J))
  SLIST(J,P)=K
ENDDO
NNB(P)=NUM
ENDDO
ENDIF
IF (NUMAX.GE.5*NNBMAX) THEN
  WRITE(ULOG,*) '   BAD H:'
  WRITE(ULOG,*) PMAX,L,NUMAX,NNB(PMAX),TNOW
  WRITE(ULOG,100) POS(PMAX,1),POS(PMAX,2),HP,RHS(PMAX)
100  FORMAT(5X,4(1PE12.5,2X))
ENDIF
IF (NFIX.NE.0) AVADJ=AVADJ/NFIX
WRITE(22,200) NMAX,BNOW,TBIN(NMP),NADVLIS,NNBSLIS,COUNT1,COUNT2
200  FORMAT(2X,7(I5,2X))
C
C  CHECK FOR OVERFLOW.
C-----
      NUM=0
      DO I=1,NNBSLIS
        P=NBSLIST(I)
        IF (INDL(P).GT.2*NNBMAX) THEN
          NUM=NUM+1
          INDX(NUM)=P
        ENDIF
      ENDDO

```

```

DO I=1,NUM
  P=INDX(I)
  WRITE(ULOG,*) P,INDL(P),POS(P,1),POS(P,2)
ENDDO
IF(NUM.NE.0) CALL ERROR(' OVERFLOW IN TLIST ')
NUM=0
DO I=1,NACC
  P=ACCLIST(I)
  IF(INDL(P).GT.2*NNBMAX) THEN
    NUM=NUM+1
    INDX(NUM)=P
  ENDIF
ENDDO
DO I=1,NUM
  P=INDX(I)
  WRITE(ULOG,*) P,INDL(P),POS(P,1),POS(P,2)
ENDDO
IF(NUM.NE.0) CALL ERROR(' OVERFLOW IN TLIST. ')
IF(COUNT1.GT.18*NNBMAX.OR.COUNT2.GT.2*NNBMAX) THEN
  DO I=1,20*NNBMAX
    P=CLIST(I)
    WRITE(ULOG,*) P,POS(P,1),POS(P,2)
  ENDDO
  WRITE(ULOG,*) COUNT1,COUNT2
  CALL ERROR(' OVERFLOW IN CLIST. ')
ENDIF
NUM=0
DO I=1,NNBSLIS
  P=NBSLIST(I)
  IF(NNB(P)+INDL(P).GT.7*NNBMAX/2) THEN
    NUM=NUM+1
    INDX(NUM)=P
  ENDIF
ENDDO
IF(NUM.NE.0) THEN
  DO I=1,NUM
    P=INDX(I)
    WRITE(ULOG,300) P,NNB(P),INDL(P),TBIN(P),BNOV
  ENDDO
  FORMAT(1X,I5,2X,5(I3,2X))
  CALL ERROR(' OVERFLOW IN SLIST. ')
ENDIF
NUM=0
DO I=1,NACC
  P=ACCLIST(I)
  IF(NNB(P)+INDL(P).GT.7*NNBMAX/2) THEN
    NUM=NUM+1
    INDX(NUM)=P

```

```

        ENDIF
    ENDDO
    IF (NUM.NE.0) THEN
        DO I=1,NUM
            P=INDX(I)
            WRITE(ULOG,300) P,NNB(P),INDL(P),TBIN(P),BNOW
        ENDDO
        CALL TERROR(' OVERFLOW IN SLIST ')
    ENDIF
CC    GOTO 35
C
C    TAKE EXTRA PARTICLES OUT OF TLISTS OF PARTICLES WHOSE H INCREASED.
C-----

    DO I=1,COUNT1
        P=CLIST(I)
        NUM=0
        DO J=1,INDL(P)
            DO 50 K=1,NNB(P)
                IF (TLIST(J,P).EQ.SLIST(K,P)) THEN
                    NUM=NUM+1
                    INDX(NUM)=J
                ENDIF
            CONTINUE
        ENDDO
        IF (NUM.GT.0) THEN
            L=1
            M=INDX(1)-1
            DO J=INDX(1),INDX(NUM)
                IF (J.EQ.INDX(L+1)) THEN
                    L=L+1
                ELSE
                    M=M+1
                    TLIST(M,P)=TLIST(J,P)
                ENDIF
            ENDDO
            DO J=INDX(NUM)+1,INDL(P)
                M=M+1
                TLIST(M,P)=TLIST(J,P)
            ENDDO
            INDL(P)=M
        ENDIF
    C    CLIST(I)=NUM
    C
    C    PUT PARTICLE INTO LIST OF NEIGHBOR IF NOT ALREADY THERE.
    C-----

        DO J=1,NNB(P)
            K=SLIST(J,P)

```

```

      IF (TBIN(K).EQ.BNOW.OR.TBIN(K).LT.0) THEN
      R=DSQRT((POS(P,1)-POS(K,1))**2+(POS(P,2)-POS(K,2))**2)
      IF (R.LT.H(K)/SOF) THEN
        DO M=1,NNB(K)
          IF (SLIST(M,K).EQ.P) GOTO 70
        ENDDO
        NNB(K)=NNB(K)+1
        SLIST(NNB(K),K)=P
70      CONTINUE
      ENDIF
      ENDIF
    ENDDO
  ENDDO
C
C  REMOVE PARTICLES IN TLIST THAT NO LONGER FIT CRITERION.
C-----

  DO I=1,COUNT2
    P=CLIST(18*NNBMAX+I)
    HP=H(P)
    NUM=0
    DO J=1,INDL(P)
      K=TLIST(J,P)
      R=DSQRT((POS(P,1)-POS(K,1))**2+(POS(P,2)-POS(K,2))**2)
      HIJ=(HP+H(K))*0.5D0
      IF (R.GT.HIJ/SOF) THEN
        NUM=NUM+1
        INDX(NUM)=J
      ENDIF
    ENDDO
    IF (NUM.GT.0) THEN
      L=1
      M=INDX(1)-1
      DO J=INDX(1),INDX(NUM)
        IF (J.EQ.INDX(L+1)) THEN
          L=L+1
        ELSE
          M=M+1
          TLIST(M,P)=TLIST(J,P)
        ENDIF
      ENDDO
      DO J=INDX(NUM)+1,INDL(P)
        M=M+1
        TLIST(M,P)=TLIST(J,P)
      ENDDO
      INDL(P)=M
    ENDIF
  C

```

C CHECK CONSISTENCY OF LISTS OF SURROUNDING PARTICLES.

```

C-----
      NUM=0
      DO J=1,NGAS
        K=ADVLIST(J)
        R=DSQRT((POS(P,1)-POS(K,1))**2+(POS(P,2)-POS(K,2))**2)
        IF(R.LT.H(K)/SOF.AND.R.GT.HP/SOF) THEN
          NUM=NUM+1
          WORKVEC(NUM)=R
          TEMPLIS(NUM)=K
        ENDIF
      ENDDO
      IF(BNOW.GE.TBIN(NMP)-2) THEN
      DO J=1,NACC
        K=ACCLIST(J)
        R=DSQRT((POS(P,1)-POS(K,1))**2+(POS(P,2)-POS(K,2))**2)
        IF(R.LT.H(K)/SOF.AND.R.GT.HP/SOF) THEN
          NUM=NUM+1
          WORKVEC(NUM)=R
          TEMPLIS(NUM)=K
        ENDIF
      ENDDO
      ENDIF
      DO J=1,NUM
        R=WORKVEC(J)
        K=TEMPLIS(J)
        HIJ=(HP+H(K))*0.5D0/SOF
        IF(R.GT.HIJ) THEN
C*VDIR: PREFER SCALAR
          DO L=1,NNB(K)
            IF(SLIST(L,K).EQ.P) THEN
              DO M=L,NNB(K)-1
                SLIST(M,K)=SLIST(M+1,K)
              ENDDO
              NNB(K)=NNB(K)-1
              GOTO 60
            ENDIF
          ENDDO
60          CONTINUE
        ELSE
          DO L=1,INDL(P)
            IF(K.EQ.TLIST(L,P)) GOTO 40
          ENDDO
          TLIST(INDL(P)+1,P)=K
          INDL(P)=INDL(P)+1
40          CONTINUE
        ENDIF
      ENDDO

```

```

        ENDDO

CC35  CONTINUE
C
C   PUT PARTICLES IN TLIST INTO SLIST.
C-----

        DO I=1,NGAS
            P=ADVLIST(I)
            IF (NNB(P)+INDL(P).GT.7*NNBMAX/2) THEN
                WRITE(ULOG,*) P,NNB(P),INDL(P)
                CALL TERROR(' OVERFLOW IN MAKELIS. ')
            ENDIF
        ENDDO
        DO I=1,NGAS
            P=ADVLIST(I)
            NUM=NNB(P)
            DO J=1,INDL(P)
                NUM=NUM+1
                SLIST(NUM,P)=TLIST(J,P)
            ENDDO
            NNB(P)=NUM
        ENDDO

C
C   BUILD LIST FROM SLIST.
C-----

        DO I=1,NGAS
            P=ADVLIST(I)
            NUM=0
            HP=H(P)
            DO J=1,NNB(P)
                K=SLIST(J,P)
                DR=DSQRT((POS(P,1)-POS(K,1))**2+(POS(P,2)-POS(K,2))**2)
                HIJ=(HP+H(K))*0.5D0
                IF(DR.LT.HIJ) THEN
                    NUM=NUM+1
                    INDX(NUM)=J
                ENDIF
            ENDDO
            IF(NUM.GT.2*NNBMAX) THEN
                WRITE(ULOG,*) P,NUM,POS(P,1),POS(P,2)
                CALL TERROR(' OVERFLOW IN LIST. ')
            ENDIF
            DO J=1,NUM
                LIST(J,P)=SLIST(INDX(J),P)
            ENDDO
        ENDDO
    
```

```

      NNBS(P)=NUM
    ENDDO
    IF(BNOW.GE.TBIN(NMP)-2) THEN
    DO I=1,NACC
      P=ACCLIST(I)
      IF(NNB(P)+INDL(P).GT.7*NNBMAX/2)
&      CALL ERROR(' OVERFLOW IN MAKELIS ')
    ENDDO
    DO I=1,NACC
      P=ACCLIST(I)
      NUM=NNB(P)
      DO J=1,INDL(P)
        NUM=NUM+1
        SLIST(NUM,P)=TLIST(J,P)
      ENDDO
      NNB(P)=NUM
    ENDDO
    DO I=1,NACC
      P=ACCLIST(I)
      NUM=0
      HP=H(P)
      DO J=1,NNB(P)
        K=SLIST(J,P)
        HIJ=(HP+H(K))*0.5D0
        DR=DSQRT((POS(P,1)-POS(K,1))**2+(POS(P,2)-POS(K,2))**2)
        IF(DR.LT.HIJ) THEN
          NUM=NUM+1
          INDX(NUM)=J
        ENDIF
      ENDDO
      IF(NUM.GT.2*NNBMAX) THEN
        WRITE(ULOG,*) P,NUM,POS(P,1),POS(P,2)
        CALL ERROR(' OVERFLOW IN LIST ')
      ENDIF
      DO J=1,NUM
        LIST(J,P)=SLIST(INDX(J),P)
      ENDDO
      NNBS(P)=NUM
    ENDDO
  ENDDIF

  RETURN
END
@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
C
      SUBROUTINE MAKETRE

```

```

C
C
C*****
C
C
C    MAIN ROUTINE TO CONTROL INITIALIZATION OF THE TREE STRUCTURE
C    FOR COMPUTING THE GRAVITATIONAL INTERACTION.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C=====

C    SET BOX PROPERTIES.
C    -----
C        CALL SETBOX
C        -----

C    LOAD BODIES INTO THE TREE.
C    -----
C        CALL LOADTRE
C        -----

C    COMPUTE PROPERTIES OF CELLS.
C    -----
C        CALL HACKCEL
C        -----

        RETURN
        END@PROCESS DC (BODYCEL, CONCOM, GAS)
C*****
C
C
C                                SUBROUTINE MAKESAP
C
C
C*****
C
C
C    MAIN ROUTINE TO CONTROL INITIALIZATION OF THE SAPLING
C    FOR FINDING NEAREST NEIGHBORS IN THE SPH CALCULATION.
C
C
C=====

```

```
/*INCLUDE NEWHEAD FORTRAN A
```

```
C=====
```

```
C  SET BOX PROPERTIES FOR SAPLING.
```

```
C  -----
```

```
C      CALL SAPBOX
```

```
C      -----
```

```
C      IF(REBOX) THEN
```

```
C  LOAD BODIES INTO THE SAPLING.
```

```
C  -----
```

```
C      CALL LOADSAP
```

```
C      -----
```

```
C  COMPUTE PROPERTIES OF CELLS.
```

```
C  -----
```

```
C      CALL HACKCEL
```

```
C      -----
```

```
C      REBOX=.FALSE.
```

```
C      ENDIF
```

```
C      RETURN
```

```
C      END
```

```
@PROCESS DC(BODYCEL, CONCOM, GAS)
```

```
C*****
```

```
C
```

```
C
```

```
          SUBROUTINE OUTASCI
```

```
C
```

```
C
```

```
C*****
```

```
C
```

```
C
```

```
C      SUBROUTINE TO OUTPUT THE BODY DATA TO AN ASCII DATA FILE.
```

```
C
```

```
C
```

```
C=====
```

```
/*INCLUDE NEWHEAD FORTRAN A
```

```
C  DECLARATION OF LOCAL VARIABLES.
```

```
C  -----
```

```
          INTEGER NDIMO, I, P, NBOD, NUM
```

```
C=====
```

C OUTPUT SYSTEM STATE.

C -----

NDIMO=NDIM

NBOD=NBODIES-NAGP+NACC

NUM=NGP-NAGP+NACC

WRITE(UBODSAS,100) NBOD,NDIMO,TNOW,NUM,NMP,QTOT

100 FORMAT(1X,2(I5,1X),G21.14,2X,2(I5,1X),G21.14)

200 FORMAT(1X,2(1PE21.14,2X))

300 FORMAT(1X,3(1PE21.14,2X))

400 FORMAT(1X,1PE21.14)

IF(NMP.EQ.1) THEN

WRITE(UBODSAS,200) POS(1,1),POS(1,2)

WRITE(UBODSAS,200) VEL(1,1),VEL(1,2)

WRITE(UBODSAS,300) MASS(1),U(1),H(1)

ENDIF

DO 10 I=1,NACC

P=ACCLIST(I)

WRITE(UBODSAS,200) POS(P,1),POS(P,2)

WRITE(UBODSAS,200) VEL(P,1),VEL(P,2)

WRITE(UBODSAS,300) MASS(P),U(P),H(P)

10 CONTINUE

C

C NOTE: THIS ASSUMES THAT NAGP HAS ONLY GAS PARTICLES.

DO I=1+NMP,NGP-NAGP

P=BODLIST(I)

WRITE(UBODSAS,200) POS(P,1),POS(P,2)

WRITE(UBODSAS,200) VEL(P,1),VEL(P,2)

WRITE(UBODSAS,300) MASS(P),U(P),H(P)

ENDDO

DO 20 I=NGP-NAGP+1,NBODLIS

P=BODLIST(I)

WRITE(UBODSAS,200) POS(P,1),POS(P,2)

WRITE(UBODSAS,200) VEL(P,1),VEL(P,2)

WRITE(UBODSAS,200) MASS(P),H(P)

20 CONTINUE

RETURN

END

@PROCESS DC(BODYCEL,CONCOM,GAS)

C*****

C

C

SUBROUTINE OUTBODS

C

C

```

C*****
C
C
C      SUBROUTINE TO OUTPUT THE BODY DATA TO BINARY OUTPUT FILE.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C
C      DECLARATION OF LOCAL VARIABLES.
C      -----

          INTEGER P,NDIMO,K

C=====
C
C      OUTPUT SYSTEM STATE.
C      -----

          NDIMO=NDIM

          WRITE(UBODSOU) NBODIES,NDIMO,TNOW,NGP,NMP,QTOT

C      THIS HAS TO BE CHANGED SO THAT P=BODLIST(I)
          WRITE(UBODSOU) (MASS(P),P=1,NBODIES),((POS(P,K),P=1,
&                      NBODIES),K=1,NDIM),((VEL(P,K),P=1,NBODIES),
&                      K=1,NDIM)
          DO 10 P=1,NGP
          WRITE(UBODSOU) U(P),H(P)
10      CONTINUE

          RETURN
          END
@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
C
C
C      SUBROUTINE OUTCPU

C
C
C*****
C
C
C      SUBROUTINE TO OUTPUT CPU TIMING DATA TO THE LOG FILE.
C
C
C

```

```

C=====

/*INCLUDE NEWHEAD FORTRAN A

C=====

C  OUTPUT TIMING DATA TO THE LOG FILE.
C  -----
      CPUTIME=CPUT1-CPUT0
      WRITE(ULOG,10) CPUTIME

10    FORMAT(//,10X,' TOTAL CPU TIME USED (SECONDS) : ',1PE12.4)

      RETURN
      END
@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
      SUBROUTINE OUTENRG
C
C
C*****
C
C
      SUBROUTINE TO OUTPUT DIAGNOSTIC DATA TO THE LOG FILE.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C  DECLARATION OF LOCAL VARIABLES.
C  -----

      DOUBLE PRECISION CPUNEW,CPUOLD,CPUSTEP,AMT,TOTM
      INTEGER K

      SAVE CPUOLD

      DATA CPUOLD/0.0/

C=====

C-----
C  WRITE MASS, ENERGY, ANGULAR MOMENTUM, CENTER OF MASS QUANTITIES.

```

```
C-----  
      WRITE(ULOG,15) MTOT,TAM  
      WRITE(ULOG,10) TAE,TAAM(3),TAEK,NAGP-NCP  
      WRITE(ULOG,20) EKTOT,EPTOT,UTOT,QTOT  
      WRITE(ULOG,40) AMVEC(1),AMVEC(2),AMVEC(3)  
      WRITE(ULOG,45) TAAM(3)+AMVEC(3),ETOT  
      WRITE(ULOG,50) (CMPOS(K),K=1,NDIM)  
      WRITE(ULOG,60) (CMVEL(K),K=1,NDIM)  
      WRITE(ULOG,65) (CMPOS(1)*CMVEL(2)-CMPOS(2)*CMVEL(1))  
  
      CALL SECONDS(CPUNEW)  
      CPUTSTEP=CPUNEW-CPUOLD  
      CPUOLD=CPUNEW  
      WRITE(ULOG,70) CPUTSTEP  
  
15     FORMAT(/,10X,'MTOT,AMTOT = ',2(1PE12.4))  
10     FORMAT(10X,'TAE,TAAM,TAEK,TNAP= ',3(1PE12.4),1X,1I5)  
20     FORMAT(10X,'EK,EP,U,Q = ',4(1PE12.4))  
40     FORMAT(10X,'AMX,AMY,AMZ = ',3(1PE12.4))  
45     FORMAT(10X,'CONS.AM,ETOT = ',2(1PE12.4))  
50     FORMAT(10X,'CMPOS = ',3(1PE12.4))  
60     FORMAT(10X,'CMVEL = ',3(1PE12.4))  
65     FORMAT(10X,'CMAM = ',1PE12.4)  
70     FORMAT(/,15X,'CPU TIME PER STEP = ',1PE12.4,/)  
  
      RETURN  
      END  
@PROCESS DC(BODYCEL,CONCOM,GAS)  
C*****  
C  
C  
          SUBROUTINE OUTERRO(MESSAGE)  
C  
C  
C*****  
C  
C  
SUBROUTINE TO OUTPUT ERROR MESSAGES TO THE LOG FILE.  
C  
C  
C=====
```

/*INCLUDE NEWHEAD FORTRAN A

DECLARATION OF LOCAL VARIABLES.

CHARACTER*(*) MESSAGE

C=====

C WRITE THE MESSAGE.

C -----

```

      WRITE(ULOG,40)
40    FORMAT(/,1X,72(' '*))
      WRITE(ULOG,50) MESSAGE
50    FORMAT(/,A)
      WRITE(ULOG,40)

```

RETURN

END

@PROCESS DC(BODYCEL,CONCOM,GAS)

C*****

C

C

SUBROUTINE OUTHEAD(OUTUNIT)

C

C

C*****

C

C

C SUBROUTINE TO OUTPUT A STANDARD HEADER TO A LOGICAL DEVICE
 C UNIT SPECIFIED BY OUTUNIT.

C

C

C=====

/*INCLUDE NEWHEAD FORTRAN A

C DECLARATION OF LOCAL VARIABLES.

C -----

INTEGER OUTUNIT

CHARACTER*3 VIND,QIND

C=====

```

      VIND='OFF'
      IF(SWITCH) VIND='ON'
      QIND='NO'
      IF(USEQUAD) QIND='YES'

```

WRITE(OUTUNIT,10)

WRITE(OUTUNIT,20)

WRITE(OUTUNIT,20)

```

WRITE(OUTUNIT,25) HEADLIN
WRITE(OUTUNIT,20)
WRITE(OUTUNIT,10)
WRITE(OUTUNIT,20)
WRITE(OUTUNIT,28)
WRITE(OUTUNIT,29)
WRITE(OUTUNIT,30) NBODIES,NGP,NMP,NSTEPS,TD
WRITE(OUTUNIT,20)
WRITE(OUTUNIT,40) CN,ARAD,EPS,TOL,ZETA
WRITE(OUTUNIT,20)
WRITE(OUTUNIT,50) ALFA,BHTA,NU,ISS,GAMMA
WRITE(OUTUNIT,20)
WRITE(OUTUNIT,60) VIND,QIND
WRITE(OUTUNIT,20)
WRITE(OUTUNIT,10)

10  FORMAT(6X,72(' '))
20  FORMAT(6X,'*',70(' '), '*')
25  FORMAT(6X,'*',10X,1A50,10X,'*')
28  FORMAT(6X,'*',4X,'INPUT PARAMETERS:',49X,'*')
29  FORMAT(6X,'*',4X,'-----',50X,'*')
30  FORMAT(6X,'*',5X,'NBODIES=',1I5,2X,'NGP=',I5,3X,'NMP=',I2,3X,
&      'NSTEPS=',1I5,2X,'TD=',1PE10.3,2X,'*')
40  FORMAT(6X,'*',5X,'CNUMB=',0PF5.2,2X,'ARAD=',0PF8.5,2X,
&      'EPS=',0PF8.5,2X,'TOL=',0PF5.2,2X,'ZETA=',0PF5.2,2X,'*')
50  FORMAT(6X,'*',5X,'ALFA=',0PF5.2,2X,'BHTA=',0PF5.2,2X,'NU=',
&      0PF6.3,2X,'ISS=',1PE11.4,2X,'GAMMA=',0PF5.2,2X,'*')
60  FORMAT(6X,'*',5X,'VISC. SWITCH=',A3,10X,'QUAD. TERMS:',A3,24X,
&      '*')

      RETURN
      END
@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
      SUBROUTINE OUTLOG(ISTEP)
C
C
C*****
C
C
C      SUBROUTINE TO MONITOR STATUS OF THE PROGRAM BY WRITING TO
C      THE LOG FILE.
C
C
C=====

```

```
/*INCLUDE NEWHEAD FORTRAN A
```

```
C  DECLARATION OF LOCAL VARIABLES.
C  -----
```

```
      INTEGER ISTEP,I,NBSMAX,NBSMIN
      DOUBLE PRECISION HMIN,HAVG,NBSAVG
```

```
C=====
```

```
C  IF FIRST CALL, WRITE HEADER.
```

```
C  -----
      IF (ISTEP.EQ.0) CALL OUTHEAD(ULOG)
C  -----
```

```
C-----
```

```
C  OUTPUT SYSTEM TIME AND FORCE EVALUATION DIAGNOSTICS.
```

```
C-----
```

```
      WRITE(ULOG,75) TNOW,ISTEP,DTS,NMAX
      WRITE(ULOG,76) TSTEP,DTIME,INCELLS
      WRITE(ULOG,80) NTTOT,NTMIN,NTMAX,NTAVG
```

```
75      FORMAT(//,7X,'TIME:',1PE11.4,2X,'STEP: ',1I4,2X,'DTS:',1PE11.4,
&          2X,'NMAX: ',I1)
76      FORMAT(24X,'TSTEP: ',I6,2X,
&          'DTIME:',1PE11.4,1X,'NCELLS: ',1I5,/)
80      FORMAT(7X,'NTTOT, MIN, MAX, AVG = ',1I8,5X,1I5,5X,1I5,5X,1I5)
```

```
      HMIN=1.D0
      HAVG=0.0
      HMAX=0.D0
      NBSAVG=0.0
      NBSMAX=0.0
      NBSMIN=3*NNBMAX
      DO I=1+NMP,NGP-NAGP
          P=BODLIST(I)
          NBSMAX=MAX(NBSMAX,NNBS(P))
          NBSMIN=MIN(NBSMIN,NNBS(P))
          NBSAVG=NBSAVG+NNBS(P)
          HMIN=MIN(HMIN,H(P))
          HMAX=MAX(HMAX,H(P))
          HAVG=HAVG+H(P)
      ENDDO
      NBSAVG=NBSAVG/DFLOAT(NGP-NMP)
      HAVG=HAVG/DFLOAT(NGP-NMP)
```

```

      WRITE(ULOG,85) HMIN,HMAX,HAVG
      WRITE(ULOG,90) NBSMIN,NBSMAX,NBSAVG
85     FORMAT(7X,'HMIN, HMAX, HAVG = ',3(E12.4,5X))
90     FORMAT(7X,'NBSMIN, NBSMAX, NBSAVG = ',2(I8,5X),E12.4)

      RETURN
      END@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
C          SUBROUTINE OUTPOS
C
C
C*****
C
C      SUBROUTINE TO OUTPUT THE POSITIONS AND VELOCITIES OF THE
C      PARTICLES FOR PLOTTING.
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C
C      DECLARATION OF LOCAL VARIABLES
C
C=====

      INTEGER I

C-----

      WRITE(15,200) NBODIES-NAGP+NACC,NDIM,TNOW,NGP-NAGP+NACC,NMP
200   FORMAT(1X,2(I5,3X),1PE12.5,3X,2(I5,3X))
      IF(NMP.NE.0) THEN
          P=NMP
          WRITE(15,100) POS(P,1),POS(P,2),VEL(P,1),VEL(P,2),MASS(P),U(P)
      ENDIF
      DO I=1,NACC
          P=ACCLIST(I)
          WRITE(15,100) POS(P,1),POS(P,2),VEL(P,1),VEL(P,2),RH(P),H(P)
      ENDDO
      DO I=1+NMP,NGP-NAGP
          P=BODLIST(I)
          WRITE(15,100) POS(P,1),POS(P,2),VEL(P,1),VEL(P,2),RH(P),H(P)
      ENDDO
      DO I=NGP+1,NBODIES
          WRITE(15,300) POS(I,1),POS(I,2),VEL(I,1),VEL(I,2)
      ENDDO

```

```

100    FORMAT(1X,6(1PE13.6,1X))

300    FORMAT(1X,4(1PE15.8,1X))
      RETURN
      END
@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
C          SUBROUTINE OUTSTAT(N)
C
C
C*****
C
C          SUBROUTINE TO OUTPUT INFORMATION ABOUT THE SYSTEM STATE TO
C          THE LOG AND BODY DATA FILES.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C  DECLARATION OF LOCAL VARIABLES.
C  -----

      INTEGER N,P,I

C=====

C          CALL OUTTERM(' STEP COMPLETED: ',N)
C          -----

      IF(N.EQ.0) THEN

          CALL OUTLOG(0)
C          -----
C          CALL OUTBODS
C          -----
          IF(TNOW.EQ.0.D0) THEN

              AMT=DSQRT(AMVEC(1)**2+AMVEC(2)**2+AMVEC(3)**2)
              WRITE(19,80) ETOT,EKTOT,EPTOT,UTOT,QTOT,AMT,TNOW

              CALL OUTDISP(0)
C          -----
C          WRITE(20,*) NACC+NGP-NAGP,BNOW,TNOW

```

```

C          IF (NMP.NE.0) THEN
C              P=NMP
C              WRITE(20,100) POS(P,1),POS(P,2),NNBS(P),NNB(P),TBIN(P)
C          ENDIF
C          DO I=1,NACC
C              P=ACCLIST(I)
C              WRITE(20,100) POS(P,1),POS(P,2),NNBS(P),NNB(P),TBIN(P)
C          ENDDO
C          DO I=1+NMP,NGP-NAGP
C              P=BODLIST(I)
C              WRITE(20,100) POS(P,1),POS(P,2),NNBS(P),NNB(P),TBIN(P)
C          ENDDO
100      FORMAT(1X,2(1PE13.6,3X),3(I3,2X))
          CALL OUTPOS
          ENDIF
          CALL ENERGY
C          -----

ELSE

          IF (TNOW.GE.TDUMP.OR.MOD(N,NOUTLOG).EQ.0) THEN

C              CALL CORRPOS('CORRECT')
C              -----

          IF (MOD(N,NOUTLOG).EQ.0) THEN

C              CALL OUTPOS
C              REWIND(UNIT=UBODSAS)
C              CALL OUTASCI
C              -----

          WRITE(20,*) NACC+NGP-NAGP,TNOW
          IF (NMP.NE.0) THEN
          WRITE(20,100) POS(NMP,1),POS(NMP,2),NNBS(NMP),TBIN(NMP)
          ENDIF
          DO I=1,NACC
              P=ACCLIST(I)
              WRITE(20,100) POS(P,1),POS(P,2),NNBS(P),TBIN(P)
          ENDDO
          DO I=1+NMP,NGP-NAGP
              P=BODLIST(I)
              WRITE(20,100) POS(P,1),POS(P,2),NNBS(P),TBIN(P)
          ENDDO

          ENDIF

          IF (TNOW.GE.TDUMP.OR.N.EQ.NSTEPS) THEN

```

```

      REWIND(UNIT=UBODSAS)
      CALL OUTASCI
C      -----
      IF(TNOW.GE.TDUMP) CALL OUTPOS
C      -----
      CALL OUTDISP(N)
C      -----
      CALL ZEROPOT
C      -----
      CALL GRAVITY('POT ')
C      -----
      CALL OUTLOG(N)
C      -----
      CALL ENERGY
C      -----
      AMT=DSQRT(AMVEC(1)**2+AMVEC(2)**2+AMVEC(3)**2)
      WRITE(19,80) ETOT,EKTOT,EPTOT,UTOT,QTOT,AMT,TNOW
80    FORMAT(1X,1PE11.4,1X,4(1PE10.3,1X),1PE11.4,1X,1PE9.2)

C      WRITE(20,*) NACC+NGP-NAGP,TNOW
C      IF(NMP.NE.0) THEN
C      WRITE(20,100) POS(NMP,1),POS(NMP,2),NNBS(NMP),TBIN(NMP)
C      ENDIF
C      DO I=1,NACC
C      P=ACCLIST(I)
C      WRITE(20,100) POS(P,1),POS(P,2),NNBS(P),TBIN(P)
C      ENDDO
C      DO I=1+NMP,NGP-NAGP
C      P=BODLIST(I)
C      WRITE(20,100) POS(P,1),POS(P,2),NNBS(P),TBIN(P)
C      ENDDO

      TDUMP=TDUMP+TD

      CALL OUTDISP(N)
C      -----
      ENDIF

      CALL CORRPOS('RESET ')
C      -----
      ENDIF

      ENDIF

      RETURN
      END
@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****

```

```

C
C
C          SUBROUTINE OUTTERM(MESSAGE,N)
C
C
C*****
C
C
C      SUBROUTINE TO OUTPUT A MESSAGE TO THE TERMINAL.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C  DECLARATION OF LOCAL VARIABLES.
C  -----

      CHARACTER*(*) MESSAGE
      INTEGER N

C=====

C  WRITE THE MESSAGE.
C  -----
C      WRITE(UTERM,*) MESSAGE,N

      RETURN
      END
@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
C          SUBROUTINE SAPBOX
C
C
C*****
C
C
C      SUBROUTINE TO BUILD LIST OF POSSIBLE NEIGHBORS, NBSLIST, TO
C      PARTICLES IN CURRENT TIME BIN (ADVLIST). RESETS SYSTEM BOX TO
C      CONTAIN ONLY PARTICLES IN NBSLIST.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

```

```

C  DECLARATION OF LOCAL VARIABLES.
C  -----

      DOUBLE PRECISION POSMIN(NDIM) , POSMAX(NDIM)
      INTEGER K, FLAG, J, NVAL, NUM, P, I, IBINS(NBINMAX+1) , NBINS

C=====

C  BUILD LIST OF POSSIBLE NEIGHBORS FROM OVERLAPPING TBINS.
C-----
      NBINS=0
      DO I=0, RMAX
        IF(RMAX(I) .GE. RMIN(BNOW) .AND. RMIN(I) .LE. RMAX(BNOW) ) THEN
          NBINS=NBINS+1
          IBINS(NBINS)=I
        ENDIF
      ENDDO
      NNBSLIS=0
      DO J=1, NBINS
        CALL WHENIEQ(NGP, TBIN, 1, IBINS(J) , TEMPLIS, NVAL)
        NUM=0
        IF(TEMPLIS(1) .EQ. NMP) NUM=1
        DO I=1+NUM, NVAL
          K=NNBSLIS+I-NUM
          NBSLIST(K)=TEMPLIS(I)
        ENDDO
        NNBSLIS=NNBSLIS+NVAL-NUM
      ENDDO

C  DETERMINE IF A RESIZING IS REQUIRED.
C  -----
      IF(NNBSLIS.LT.NBODLIS) REBOX=.TRUE.
CCC IF I'M NOT BUILDING A TREE FOR GRAVITY:
CCC   REBOX=.TRUE.
      IF(REBOX) THEN
C  DETERMINE MINIMUM AND MAXIMUM COORDINATES OF BODIES.
C  -----

      DO I=1, NDIM
        POSMIN(I)=100.
        POSMAX(I)=-100.
      ENDDO
      DO K=1, NDIM
        DO I=1, NNBSLIS
          P=NBSLIST(I)
          POSMAX(K)=MAX(POSMAX(K) , POS(P, K) )
          POSMIN(K)=MIN(POSMIN(K) , POS(P, K) )
        ENDDO
      ENDDO

```

```

      ENDDO

C   IF A RESIZING IS NECESSARY, RECOMPUTE RSIZE AND RMIN.
C   -----

      RSIZE=0.D0
      DO 70 K=1,NDIM
        RSIZE=MAX(RSIZE, POSMAX(K)-POSMIN(K))
70    CONTINUE

      DO 80 K=1,NDIM
        RBMIN(K)=0.5D0*(POSMIN(K)+POSMAX(K))-0.5D0*RSIZE
80    CONTINUE
      ENDIF
      RETURN
      END

@PROCESS DC(BODYCEL, CONCOM, GAS)
C*****
C
C
C               SUBROUTINE SETBOX
C
C
C*****
C
C   SUBROUTINE TO ADJUST SYSTEM BOX SO THAT IT CONTAINS ALL BODIES.
C   THE LOCAL VARIABLE REBOX INDICATES WHETHER A RESIZING OF THE
C   SYSTEM BOX IS TO TAKE PLACE. THE VARIABLES POSMIN AND POSMAX
C   ARE THE MINIMUM AND MAXIMUM COORDINATES OF BODIES IN EACH
C   DIMENSION.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C   DECLARATION OF LOCAL VARIABLES.
C   -----

      DOUBLE PRECISION POSMIN(NDIM), POSMAX(NDIM), POSX(NBODSMA),
&      POSY(NBODSMA), POSZ(NBODSMA)
      INTEGER K, ISMIN, ISMAX, I

      EQUIVALENCE (POSX(1), POS(1,1)), (POSY(1), POS(1,2))

C=====

```

```

C      DETERMINE MINIMUM AND MAXIMUM COORDINATES OF BODIES.
C      -----
C      POSMIN(1)=POSX(ISMIN(NBODIES, POSX,1))
C      POSMIN(2)=POSY(ISMIN(NBODIES, POSY,1))
C      POSMIN(3)=POSZ(ISMIN(NBODIES, POSZ,1))
C      POSMAX(1)=POSX(ISMAX(NBODIES, POSX,1))
C      POSMAX(2)=POSY(ISMAX(NBODIES, POSY,1))
C      POSMAX(3)=POSZ(ISMAX(NBODIES, POSZ,1))
C
C      DETERMINE IF A RESIZING IS REQUIRED.
C      -----
C      DO 50 K=1,NDIM
C          IF(RBMIN(K).GT.POSMIN(K).OR.RBMIN(K)+RSIZE.LT.POSMAX(K).
50      &      OR.NBBSLIS.LT.NBODLIS) REBOX=.TRUE.
C      CONTINUE
C
C      IF A RESIZING IS NECESSARY, RECOMPUTE RSIZE AND RMIN.
C      -----
C
C      IF(REBOX) THEN
C          DO 70 K=1,NDIM
C              RSIZE=MAX(RSIZE, POSMAX(K)-POSMIN(K))
70      CONTINUE
C
C          DO 80 K=1,NDIM
C              RBMIN(K)=0.5D0*(POSMIN(K)+POSMAX(K))-0.5D0*RSIZE
80      CONTINUE
C
C      ENDIF
C
C      REBOX=.FALSE.
C
C      RETURN
C      END
C
C@PROCESS DC(BODYCEL, CONCOM, GAS)
C*****
C
C
C          SUBROUTINE STARTOU
C
C
C*****
C
C
C      SUBROUTINE TO INITIALIZE DISK FILES FOR SUBSEQUENT INPUT/OUTPUT.
C      ALL FILES, OTHER THAN THE BINARY BODY DATA FILE, ARE ASSUMED TO
C      BE OF ASCII (TEXT) FORM.

```

```

C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C=====

C  OPEN PARAMETER FILE.
C  -----
      OPEN(UNIT=UPARS)

C  OPEN LOG FILE.
C  -----
      OPEN(UNIT=ULOG)

C  OPEN INPUT BODY DATA FILE.
C  -----
      OPEN(UNIT=UBODSIN)

C  OPEN OUTPUT BINARY BODY DATA FILE.
C  -----
      OPEN(UNIT=UBODSOU)

C  OPEN OUTPUT ASCII BODY DATA FILE.
C  -----
      OPEN(UNIT=UBODSAS)

C  OPEN OUTPUT POSITION DATA FILE.
C  -----
      OPEN(UNIT=15)

C  OPEN OUTPUT VELOCITY DATA FILE.
C  -----
      OPEN(UNIT=16)

C  OPEN OUTPUT MASS DATA FILE.
C  -----
      OPEN(UNIT=17)

C  OPEN OUTPUT PROCTOSTAR DATA FILE
C  -----
      OPEN(UNIT=13)

C  OPEN OUTPUT ENERGY DATA FILE.
C  -----
      OPEN(UNIT=19)

```

```
C  OPEN OUTPUT NUMBER DENSITY DATA FILE
```

```
C  -----
      OPEN(UNIT=20)
```

```
C  OPEN OUTPUT DENSITY AT GRID POINTS FILE.
```

```
C  -----
      OPEN(UNIT=21)
```

```
C  OPEN OUTPUT BIN DATA FILE.
```

```
C  -----
      OPEN(UNIT=22)
```

```
      RETURN
```

```
      END
```

```
@PROCESS DC (BODYCEL, CONCOM, GAS)
```

```
@PROCESS DIRECTIVE(' *VDIR:')
```

```
C*****
```

```
C
```

```
C
```

```
      SUBROUTINE STEPPOS
```

```
C
```

```
C
```

```
C*****
```

```
C
```

```
C
```

```
C      SUBROUTINE TO ADVANCE THE POSITIONS OF THE BODIES FOR A
C      TIMESTEP DTIME/2.
```

```
C
```

```
C
```

```
C=====
```

```
/*INCLUDE NEWHEAD FORTRAN A
```

```
C  DECLARATION OF LOCAL VARIABLES.
```

```
C  -----
```

```
      INTEGER P,K,I
```

```
C=====
```

```
C  LOOP OVER ALL SPATIAL COORDINATES FOR ALL BODIES.
```

```
C  -----
```

```
      DO 200 K=1,NDIM
```

```
C*VDIR: IGNORE RECRDEPS
```

```
      DO 100 I=1,NBODLIS
```

```
      P=BODLIST(I)
```

```

                POS(P,K)=POS(P,K)+VEL(P,K)*DTIME2
100      CONTINUE
200      CONTINUE
C
C      STEP ACCRETED PARTICLES WITH P=1
C-----
        DO K=1,NDIM
C*VDIR: IGNORE RECRDEPS
            DO I=1,NACC
                P=ACCLIST(I)
                POS(P,K)=POS(P,K)+VEL(1,K)*DTIME2
            ENDDO
        ENDDO

C      UPDATE POSITION TIME, SYSTEM TIME.
C      -----
        TPOS=TPOS+DTIME2
        TELAPS=TELAPS+DTIME2
        TNOW=TPOS

        RETURN
        END

@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
C              SUBROUTINE STEPSYS(N)
C
C
C*****
C
C      SUBROUTINE TO ADVANCE THE STATE OF THE SYSTEM BY ONE LARGE
C      TIMESTEP.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C      DECLARATION OF LOCAL VARIABLES.
C      -----

        INTEGER N

C=====

```

```

C   UPDATE POSITIONS BY 1/2 STEP.
C   -----
      IF (TELAPS.NE.0.D0.AND.NADVLIS.NE.0) THEN
        CALL HALFSTP(N)
C   -----
      ELSE
        IF (TELAPS.NE.0.D0) THEN
          DO K=1,NDIM
            DO I=1,NBODLIS
              P=BODLIST(I)
              VET(P,K)=VEL(P,K)
            ENDDO
          ENDDO
        ENDIF

        CALL ESTVEL
C   -----
        CALL STEPPOS
C   -----
      ENDIF

C   MAKE LIST OF PARTICLES TO BE ADVANCED IN THE NEXT HALF STEP.
C   -----
      BNOW=NMAX
      CALL WHENIEQ(NBODIES,TBIN,1,BNOW,ADVLIST,NADVLIS)
C   -----
      NGAS=0
      I=0
      DOWHILE(ADVLIST(I+1).LE.NGP.AND.(I+1).LE.NADVLIS)
        I=I+1
      ENDDO
      NGAS=I

      CALL HALFSTP(N)
C   -----
      CALL AMFLUX
C   -----
      CALL ACCRET(N)
C   -----
      CALL COURANT(N)
C   -----

      RETURN
      END
@PROCESS DC(BODYCEL,CONCOM,GAS)
@PROCESS DIRECTIVE(' *VDIR:')
C*****
C

```

```

C
C                                     SUBROUTINE STEPVEL
C
C*****
C
C
C      SUBROUTINE TO ADVANCE THE VELOCITIES OF THE BODIES FOR A
C      TIMESTEP DTIME.
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C  DECLARATION OF LOCAL VARIABLES.
C  -----

      INTEGER P,K,I
      DOUBLE PRECISION DTNOW

C=====

C  LOOP OVER ALL VELOCITY COMPONENTS FOR ALL BODIES.
C  -----

      DTNOW=DTS/2.D0**BNOW
      DO 200 K=1,NDIM
C*VDIR: IGNORE RECRDEPS
          DO 100 I=1,NADVLIS
              P=ADVLIST(I)
              VEL(P,K)=VEL(P,K)+ACC(P,K)*DTNOW
              VET(P,K)=VEL(P,K)-ACC(P,K)*DTNOW*0.5D0
          100      CONTINUE
      200      CONTINUE

C  EVOLVE INTERNAL ENERGIES.
C-----
      DO I=1,NADVLIS
          P=ADVLIST(I)
          U(P)=U(P)+UDOT(P)*DTNOW
      ENDDO

      RETURN
      END@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C

```

```

C
C                               SUBROUTINE STOPOUT
C
C
C*****
C
C
C      SUBROUTINE TO CLOSE THE OUTPUT FILES.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C=====

C      CLOSE THE OPEN FILES.
C      -----
C          CLOSE(UNIT=UBODSOU)
C          CLOSE(UNIT=UBODSAS)
C          CLOSE(UNIT=ULOG)

C          RETURN
C          END
@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
C                               SUBROUTINE TERROR(MESSAGE)
C
C
C*****
C
C
C      SUBROUTINE TO TERMINATE THE PROGRAM AS THE RESULT OF A FATAL
C      ERROR, CLOSE THE OUTPUT FILES, AND DUMP TIMING INFORMATION.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C      DECLARATION OF LOCAL VARIABLES.
C      -----

C          CHARACTER*(*) MESSAGE

```

```

C=====
C  WRITE ERROR MESSAGE TO THE LOG FILE.
C  -----
C      CALL OUTERRO(MESSAGE)
C      -----
C          WRITE(20,*) NACC+NGP-NAGP,TNOW
C          IF(NMP.NE.0) THEN
C              WRITE(20,100) POS(NMP,1),POS(NMP,2),NNBS(NMP),TBIN(NMP)
C          ENDIF
C          DO I=1,NACC
C              P=ACCLIST(I)
C              WRITE(20,100) POS(P,1),POS(P,2),NNBS(P),TBIN(P)
C          ENDDO
C          DO I=1+NMP,NGP-NAGP
C              P=BODLIST(I)
C              WRITE(20,100) POS(P,1),POS(P,2),NNBS(P),TBIN(P)
C          ENDDO
100      FORMAT(1X,2(1PE13.6,3X),2(I3,3X))

C  OUTPUT SYSTEM STATE.
C  -----
C      CALL CORRPOS('CORRECT')
C      -----
C      CALL ZEROPOT
C      -----
C      CALL GRAVITY('POT ')
C      -----
C      CALL OUTASCI
C      -----
C      CALL OUTBODS
C      -----

C-----
C  STOP TIMING, OUTPUT TIMING DATA, CLOSE FILES, TERMINATE THE
C  SIMULATION.
C-----

C      CALL SECONDS(CPUT1)
C      -----
C      CALL OUTCPU
C      -----
C      CALL STOPOUT
C      -----

C      STOP
C      END
@PROCESS DC(BODYCEL,CONCOM,GAS)

```

```

C*****
C
C
C          SUBROUTINE TOLTEST
C
C
C*****
C
C
C      SUBROUTINE TO TERMINATE THE PROGRAM AS THE RESULT OF A FATAL
C      ERROR, CLOSE THE OUTPUT FILES, AND DUMP TIMING INFORMATION.
C
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

      INTEGER P,I,J,NTERMS,SMINDEX,NTOL,K
      PARAMETER (NTOL=5)
      DOUBLE PRECISION TOL0,DX,DY,DRDOTDR,SDRDOTD,RINVEFF,R3INVEF,
&          EPSI,DRDELD,DRSM,EPSP,ACCSM,ACCI,PMASS,
&          DEVX,DEVY,TOLER(NTOL)
      DATA TOLER/0.4D0,0.5D0,0.6D0,0.7D0,0.8D0/

      WRITE(22,*) NBODLIS
      DO I=1,NBODLIS
        P=BODLIST(I)
        PMASS=MASS(P)
        MASS(P)=0.D0
        EPSP=EPS
        EPSI=EPS
      DO J=1,NBODLIS
        K=BODLIST(J)
        DX=POS(P,1)-POS(K,1)
        DY=POS(P,2)-POS(K,2)
        DRDOTDR=DX**2+DY**2
        SDRDOTD=SQRT(DRDOTDR)
        RINVEFF=1./(SDRDOTD+TINY)
        R3INVEF=RINVEFF/(DRDOTDR+TINY)
        DRDELD=SDRDOTD*NINTERP/(EPSP+EPSI)
        SMINDEX=DRDELD
        SMINDEX=MIN(NINTERP,SMINDEX)
        DRSM=MIN(ONE,DRDELD-SMINDEX)
        ACCSM=(1.-DRSM)*ACSMOOT(SMINDEX)+DRSM*ACSMOOT(1+SMINDEX)
        R3INVEF=ACCSM*R3INVEF
        ACCI=MASS(K)*R3INVEF
        ACC(P,1)=ACC(P,1)-DX*ACCI
        ACC(P,2)=ACC(P,2)-DY*ACCI

```

```

      ENDDO
      MASS(P)=PMASS
      WRITE(22,*) POS(P,1),POS(P,2),ACC(P,1),ACC(P,2)
      VEL(P,1)=ACC(P,1)
      VEL(P,2)=ACC(P,2)
      ACC(P,1)=0.D0
      ACC(P,2)=0.D0
      ENDDO

      DO J=1,NTOL
        TOL2INV=1.D0/TOLER(J)**2
        DO I=1,NBODLIS
          P=BODLIST(I)
          CALL TREEWAL(P,NTERMS)
          CALL GRAVSUM(P,NTERMS,'ACC ')
          DEVX=ACC(P,1)-VEL(P,1)
          DEVY=ACC(P,2)-VEL(P,2)
          WRITE(22,*) DEVX,DEVY
          ACC(P,1)=0.D0
          ACC(P,2)=0.D0
        ENDDO
      ENDDO

      TOL2INV=1.D0/TOL**2

      RETURN
      END
      @PROCESS DC(BODYCEL,CONCOM,GAS) DIRECTIVE('*VDIR:')
      C*****
      C
      C
      C          SUBROUTINE TREEWAL(P,NTERMS)
      C
      C
      C*****
      C
      C
      C          SUBROUTINE TO WALK THROUGH THE TREE AND ACCUMULATE THE LIST OF
      C          INTERACTIONS FOR BODY P.  THE INTERACTION LIST IS PASSED BACK
      C          TO THE CALLING SUBROUTINE ACCGRAV IN THE VECTOR ITERM, WHICH
      C          IS EQUIVALENCED TO THE COMMON ARRAY ACTLIST.  VECTORIZATION IS
      C          ACHIEVED BY PROCESSING ALL CELLS AT THE SAME LEVEL IN THE TREE
      C          SIMULTANEOUSLY (HERNQUIST, J. COMPUT. PHYS., SUBMITTED [1988]).
      C
      C
      C=====

      /*INCLUDE NEWHEAD FORTRAN A

```

```

C  DECLARATION OF LOCAL VARIABLES.
C  -----

      DOUBLE PRECISION CVMGT,TESTKEE
      INTEGER P,I,NNODES,NKEEP,NSUBDIV,NTERMS,ITERMS(NBODSMA),
&          NODELIS(NBODSMA),KEEPER(NBODSMA)
      LOGICAL TOLCRIT

      EQUIVALENCE (ITERMS(1),ACTLIST(1)),(NODELIS(1),CELLIST(1)),
&          (KEEPER(1),PARENT(1))

C=====

C  INITIALIZE LIST OF CELLS TO EXAMINE.
C  -----
      NTERMS=0
      NNODES=1
      NODELIS(1)=ROOT

10    CONTINUE

C  LOOP UNTIL NO CELLS ARE LEFT TO EXAMINE.
C  -----
      IF(NNODES.GT.0) THEN

C  APPLY TOLERANCE CRITERION TO LIST OF CELLS.
C  -----

      DO 20 I=1,NNODES
          TOLCRIT=(POS(P,1)-POS(NODELIS(I),1))**2+(POS(P,2)-
&          POS(NODELIS(I),2))**2.GE.
&          (CELLSIZ(NODELIS(I))**2*TOL2INV)
          TESTKEE=CVMGT(TWO,MINUST,TOLCRIT)
          KEEPER(I)=INT(TESTKEE)
20    CONTINUE

C-----
C  ADD CELLS WHICH SATISFY CRITERION TO INTERACTION LIST.  NOTE THAT,
C  DEPENDING ON THETA, SELF-INTERACTION TERM WILL BE INCLUDED.
C-----

      CALL WHENIGT(NNODES,KEEPER,1,0,ISUBSET,NKEEP)

      IF(NTERMS+NKEEP.GT.NBODSMA)
&          CALL TERROR(' ARRAY OVERFLOW IN TREEWAL ')
C          -----

      DO 30 I=1,NKEEP

```

```

      1TERMS (NTERMS+1) = NODELIS (ISUBSET (I))
30      CONTINUE

      NTERMS = NTERMS + NKEEP

C-----
C      ADD SUBCELLS OF CELLS WHICH FAIL TOLERANCE CRITERION TO LIST OF
C      CELLS TO EXAMINE.
C-----

      CALL WHENILT (NNODES, KEEPTER, 1, 0, ISUBSET, NSUBDIV)

      IF (4*NSUBDIV.GT.NBODSMA.OR.4*NSUBDIV.GT.NCELLS)
&          CALL TERROR (' ASUBP OVERFLOW IN TREEWAL ')
C          -----

C*VDIR: IGNORE RECRDEPS
      DO 40 I=1, NSUBDIV
          ASUBP (I) = SUBP (NODELIS (ISUBSET (I)), 1)
          ASUBP (I+NSUBDIV) = SUBP (NODELIS (ISUBSET (I)), 2)
          ASUBP (I+2*NSUBDIV) = SUBP (NODELIS (ISUBSET (I)), 3)
          ASUBP (I+3*NSUBDIV) = SUBP (NODELIS (ISUBSET (I)), 4)
C      NEED FOLLOWING FOR 3D.
C          ASUBP (I+4*NSUBDIV) = SUBP (NODELIS (ISUBSET (I)), 5)
C          ASUBP (I+5*NSUBDIV) = SUBP (NODELIS (ISUBSET (I)), 6)
C          ASUBP (I+6*NSUBDIV) = SUBP (NODELIS (ISUBSET (I)), 7)
C          ASUBP (I+7*NSUBDIV) = SUBP (NODELIS (ISUBSET (I)), 8)
40      CONTINUE

      CALL WHENNE (4*NSUBDIV, ASUBP, 1, 0, ISUBSET, NNODES)

      DO 60 I=1, NNODES
          NODELIS (I) = ASUBP (ISUBSET (I))
60      CONTINUE

      GO TO 10

ENDIF

RETURN
END

@PROCESS DC (BODYCEL, CONCOM, GAS) DIRECTIVE ('*VDIR:')
C*****
C
C
      SUBROUTINE WALKIT (P, HP, NTERMS)
C
C

```

```

C*****
C
C
C      SUBROUTINE TO WALK THROUGH THE TREE AND ACCUMULATE THE LIST OF
C      NEIGHBORS FOR BODY P.  THIS LIST OF NEIGHBORS IS PASSED BACK
C      TO THE CALLING SUBROUTINE HYDRO IN THE VECTOR TEMPLIS AND THE
C      DISTANCES ARE PASSED IN VECTOR WORKVEC.  VECTORIZATION IS
C      ACHIEVED BY PROCESSING ALL CELLS AT THE SAME LEVEL IN THE TREE
C      SIMULTANEOUSLY.  THIS SUBROUTINE IS ADAPTED FROM TREEWAL.
C      R. DRIMMEL 1991.
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C  DECLARATION OF LOCAL VARIABLES.
C  -----

      DOUBLE PRECISION CUMGT,TESTKEE,RTWO,DR,HP
      INTEGER P,I,K,NNODES,NKEEP,NSUBDIV,NTERMS,
&      NODELIS(NBODSMA),KEEPER(NBODSMA)
      LOGICAL TOLCRIT

C=====

      RTWO=DSQRT(TWO)

C  INITIALIZE LIST OF CELLS TO EXAMINE.
C  -----
      NTERMS=0
      NNODES=1
      NODELIS(1)=ROOT

10  CONTINUE

C  LOOP UNTIL NO CELLS ARE LEFT TO EXAMINE.
C  -----
      IF(NNODES.GT.0) THEN

C  APPLY TOLERANCE CRITERION TO LIST OF CELLS.
C  -----

      DO 20 I=1,NNODES
          TOLCRIT= DSQRT((POS(P,1)-POS(NODELIS(I),1))**2+
&      (POS(P,2)-POS(NODELIS(I),2))**2)
&      -RTWO*CELLSIZ(NODELIS(I)).GE.HP

```

```

TESTKEE=CVMGT(1,NO,MINUST,TOLCRIT)
KEEPER(I)=INT(TESTKEE)
20      CONTINUE

C-----
C  CELLS WHICH SATISFY CRITERION ARE IGNORED.
C  ADD SUBCELLS OF CELLS WHICH FAIL TOLERANCE CRITERION TO LIST OF
C  CELLS TO EXAMINE. IF SUBCELLS ARE LEAVES THAT CONTAIN A GAS PARTICLE
C  THEY ARE ADDED TO INTERACTION LIST
C-----

      CALL WHENILT(NNODES,KEEPER,1,0,ISUBSET,NSUBDIV)

      IF(4*NSUBDIV.GT.NBODSMA.OR.4*NSUBDIV.GT.NCELLS)
&        CALL TERROR(' ASUBP OVERFLOW IN WALKIT ')
C        -----

C*VDIR: IGNORE RECRDEPS
      DO 40 I=1,NSUBDIV
        ASUBP(I)=SUBP(NODELIS(ISUBSET(I)),1)
        ASUBP(I+NSUBDIV)=SUBP(NODELIS(ISUBSET(I)),2)
        ASUBP(I+2*NSUBDIV)=SUBP(NODELIS(ISUBSET(I)),3)
        ASUBP(I+3*NSUBDIV)=SUBP(NODELIS(ISUBSET(I)),4)
C      NEED FOLLOWING FOR 3D:
C        ASUBP(I+4*NSUBDIV)=SUBP(NODELIS(ISUBSET(I)),5)
C        ASUBP(I+5*NSUBDIV)=SUBP(NODELIS(ISUBSET(I)),6)
C        ASUBP(I+6*NSUBDIV)=SUBP(NODELIS(ISUBSET(I)),7)
C        ASUBP(I+7*NSUBDIV)=SUBP(NODELIS(ISUBSET(I)),8)
40      CONTINUE

      CALL WHENILE(4*NSUBDIV,ASUBP,1,NGP,ISUBSET,NNODES)

      IF(NTERMS+NNODES.GT.NBODSMA)
&        CALL TERROR(' ARRAY OVERFLOW IN WALKIT')

      DO I=1,NNODES
        KEEPER(I)=ASUBP(ISUBSET(I))
      ENDDO

      CALL WHENIGT(NNODES,KEEPER,1,0,ISUBSET,NKEEP)

      DO I=1,NKEEP
        K=KEEPER(ISUBSET(I))
        DR=DSQRT((POS(K,1)-POS(P,1))**2+(POS(K,2)-POS(P,2))**2)
        IF(DR.LT.HP.AND.K.NE.P) THEN
          NTERMS=NTERMS+1
          TEMPLIS(NTERMS)=K
          WORKVEC(NTERMS)=DR

```

```

ENDIF
ENDDO

CALL WHENIGT(4*NSUBDIV,ASUBP,1,NBODSMA,ISUBSET,NNODES)

DO 60 I=1,NNODES
    NODELIS(I)=ASUEP(ISUBSET(I))
CONTINUE

GO TO 10

ENDIF

C
C SEARCH THROUGH ACCRETED PARTICLES
C-----
IF(NACC.NE.0.AND.P.NE.NMP) THEN
    DR=DSQRT((POS(NMP,1)-POS(P,1))**2+(POS(NMP,2)-POS(P,2))**2)
    IF(DR.LT.(ARAD+HP)) THEN
        DO I=1,NACC
            K=ACCLIST(I)
            DR=DSQRT((POS(K,1)-POS(P,1))**2+(POS(K,2)-POS(P,2))**2)
            IF(DR.LT.HP.AND.K.NE.P) THEN
                NTERMS=NTERMS+1
                TEMPLIS(NTERMS)=K
                WORKVEC(NTERMS)=DR
            ENDIF
        ENDDO
    ENDIF
ENDIF
RETURN
END

@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
C
C SUBROUTINE ZEROACC
C
C *****
C
C SUBROUTINE TO ZERO OUT ACCELERATION.
C=====

```

```
/*INCLUDE NEWHEAD FORTRAN
```

```
C  DECLARATION OF LOCAL VARIABLES.
C  -----
```

```
      INTEGER P
```

```
C=====
```

```
      DO 10 I=1,NADVLIS
        P=ADVLIST(I)
        ACC(P,1)=0.
        ACC(P,2)=0.
C        ACC(P,3)=0.
10      CONTINUE
```

```
      RETURN
      END
```

```
@PROCESS DC(BODYCEL,CONCOM GAS)
```

```
C*****
C
C
```

```
      SUBROUTINE ZEROPOT
```

```
C
C
C*****
C
C
C      SUBROUTINE TO ZERO OUT POTENTIAL.
C
C
C=====
```

```
/*INCLUDE NEWHEAD FORTRAN A
```

```
C  DECLARATION OF LOCAL VARIABLES.
C  -----
```

```
      INTEGER P
```

```
C=====
```

```
      DO 10 P=1,NBODIES
        PHI(P)=0.
10      CONTINUE
```

RETURN
END

```
C*****  
C  
C  
  
C          SUBROUTINE RANSET(SD)  
C  
C  
C*****  
C  
C  
C  
C      DUMMY SUBROUTINE TO INITIALIZE RANDOM NUMBERS.  
C  
C  
C=====
```

DOUBLE PRECISION SE

RETURN
END

```
C*****  
C  
C  
C          SUBROUTINE SECONDS(CPU)  
C  
C  
C*****  
C  
C  
C          SUBROUTINE TO RETURN ELAPSED CPU TIME.  
C  
C=====
```

DOUBLE PRECISION CPU
INTEGER NCODE

```
CALL CPUTIME(CPU,NCODE)
```

CPU=CPU/1.0D06

RETURN
END

```

C
C
C          FUNCTION CVMGP(Y1,Y2,Y3)
C
C*****
C
C
C          FUNCTION TO PERFORM CONDITIONAL VECTOR MERGE ON POSITIVE
C          REQUIRED BY CRAY VERSION.
C
C=====

      DOUBLE PRECISION CVMGP,Y1,Y2,Y3

      IF(Y3.GE.0.0) THEN
        CVMGP=Y1
      ELSE
        CVMGP=Y2
      ENDIF

      RETURN
      END

C*****
C
C          FUNCTION CVMGT(Y1,Y2,Y3)
C
C*****
C
C          FUNCTION TO PERFORM CONDITIONAL VECTOR MERGE ON TRUE REQUIRED
C          BY CRAY VERSION.
C
C=====

      DOUBLE PRECISION CVMGT,Y1,Y2
      LOGICAL Y3

      IF(Y3) THEN
        CVMGT=Y1
      ELSE
        CVMGT=Y2
      ENDIF

```

```

RETURN
END

```

```

C*****
C
C

```

```

FUNCTION ISMAX(N,X,INC)

```

```

C
C
C*****
C
C

```

```

C    FUNCTION TO LOCATE INDEX OF MAXIMUM ELEMENT OF A REAL VECTOR.
C
C

```

```

C=====

```

```

DOUBLE PRECISION X(1),XMAX
INTEGER ISMAX,N,INC,I

```

```

ISMAX=1
XMAX=X(1)

```

```

DO 10 I=2,N,INC
  IF (X(I) .GT. XMAX) THEN
    ISMAX=I
    XMAX=X(I)
  ENDIF

```

```

10  CONTINUE

```

```

RETURN
END

```

```

C*****
C
C

```

```

FUNCTION ISMIN(N,X,INC)

```

```

C
C
C*****
C
C

```

```

C    FUNCTION TO LOCATE INDEX OF MINIMUM ELEMENT OF A REAL VECTOR.
C
C

```

```

C=====

```

```

DOUBLE PRECISION X(1),XMIN

```

```

      INTEGER ISMIN,N,INC,I

      ISMIN=1
      XMIN=X(1)

      DO 10 I=2,N,INC
        IF(X(I).LT.XMIN) THEN
          ISMIN=I
          XMIN=X(I)
        ENDIF
10    CONTINUE

      RETURN
      ENDC*****

C
C
      FUNCTION ISRCHIG(N,IARRAY,INC,ITARGET)

C
C
C*****
C
C
C      FUNCTION TO RETURN INDEX OF FIRST ELEMENT OF IARRAY GREATER
C      THAN ITARGET, OR N+1 IF NONE IS FOUND.
C
C
C=====

      INTEGER ISRCHIG,N,IARRAY(1),INC,ITARGET,I

      ISRCHIG=N+1

      DO 10 I=1,N,INC
        IF(IARRAY(I).GT.ITARGET) THEN
          ISRCHIG=I
          RETURN
        ENDIF
10    CONTINUE

      RETURN
      END
C*****
C
      FUNCTION RAN3(IDUM)

C
C-----
C
C      Returns a uniform random deviate between 0.0 and 1.0.  Set IDUM

```

C to any negative value to initialize or reinitialize the sequence.
 C If running on machine that is weak on integer arithmetic, then use
 C commented lines. Any other large value of MBIG and MSED will
 C work. This routine is originally from Knuth, but the current
 C version is from Numerical Recipes by Press et al.

C
 C=====

```

      IMPLICIT NONE
      DOUBLE PRECISION  FAC,RAN3
      INTEGER  MBIG,MSEED,IFF,IDUM,MJ,MA(55),MK,II,MZ,K,I,
&  INEXT,INEXTP
      PARAMETER (MBIG=1000000000,MSEED=161803398,MZ=0,FAC=1./MBIG)
      SAVE IFF,MA,MZ,INEXT,INEXTP
      DATA IFF/0/

```

```

      IF (IDUM.LT.0.OR.IFF.EQ.0) THEN
        IFF=1
        MJ=MSEED-IABS(IDUM)
        MJ=MOD(MJ,MBIG)
        MA(55)=MJ
        MK=1
        DO I=1,64
          II=MOD(21*I,55)
          MA(II)=MK
          MK=MJ-MK
          IF (MK.LT.MZ) MK=MK+MBIG
          MJ=MA(II)
        ENDDO
        DO K=1,4
          DO I=1,55
            MA(I)=MA(I)-MA(1+MOD(I+30,55))
            IF (MA(I).LT.MZ) MA(I)=MA(I)+MBIG
          ENDDO
        ENDDO
        INEXT=0
        INEXTP=31
        IDUM=1
      ENDIF
      INEXT=INEXT+1
      IF (INEXT.EQ.56) INEXT=1
      INEXTP=INEXTP+1
      IF (INEXTP.EQ.56) INEXTP=1
      MJ=MA(INEXT)-MA(INEXTP)
      IF (MJ.LT.MZ) MJ=MJ+MBIG
      MA(INEXT)=MJ
      RAN3=MJ*FAC
      RETURN
      END

```

```

C*****
C
C
C          SUBROUTINE LINK(DUMMESS)
C
C
C*****
C
C
C          DUMMY SUBROUTINE TO HANDLE CALL TO CRAY LINKER.
C
C
C=====

          CHARACTER*(*) DUMMESS

          RETURN
          END

C*****
C
C
C          SUBROUTINE PKSRT(N,ARR,INDX)
C
C
C*****
C
C          SUBROUTINE TO SORT ARRAY ARR AND OUTPUT THE ARRAY OF INDICES
C          INDX IN ASCENDING ORDER.  ADDAPTED FROM ROUTINE IN NUMERICAL
C          RECIPES.
C
C=====
          INTEGER  I,J,INDX,N
          DOUBLE PRECISION  ARR,A
          DIMENSION INDX(N),ARR(N)
          DO I=1,N
             INDX(I)=I
          ENDDO
          DO J=2,N
             A=ARR(J)
             DO I=J-1,1,-1
                IF(ARR(INDX(I)).LE.A) GOTO 10
                INDX(I+1)=INDX(I)
             ENDDO
             I=0
10          INDX(I+1)=J
          ENDDO

```

```

      RETURN
      END
C*****
C
C
C                               SUBROUTINE NPKSRT(N,ARR,INDX)
C
C*****
C
C      SUBROUTINE TO SORT ARRAY ARR AND OUTPUT THE ARRAY OF INDICES
C      INDX IN ASCENDING ORDER.  ADAPTED FROM ROUTINE IN NUMERICAL
C      RECIPES.
C
C=====
C      INTEGER  I,J,INDX,N,ARR,A
C      DIMENSION INDX(N),ARR(N)
C      DO I=1,N
C         INDX(I)=I
C      ENDDO
C      DO J=2,N
C         A=ARR(J)
C         DO I=J-1,1,-1
C            IF (ARR(INDX(I)).LE.A) GOTO 10
C            INDX(I+1)=INDX(I)
C         ENDDO
C         I=0
C10      INDX(I+1)=J
C      ENDDO
C
C      RETURN
C      END
C*****
C
C
C                               SUBROUTINE WHENEQ(N,IARRAY,INC,ITARGET,INDEX,NVAL)
C
C*****
C
C
C      SUBROUTINE TO RETURN LOCATIONS OF ALL ELEMENTS OF AN INTEGER
C      VECTOR EQUAL TO ITARGET.
C
C=====
C
C      INTEGER IARRAY(1),INDEX(1),N,INC,ITARGET,NVAL,I

```

```

      NVAL=0

      DO 10 I=1,N,INC
        IF (IARRAY(I) .EQ. ITARGET) THEN
          NVAL=NVAL+1
          INDEX(NVAL)=I
        ENDIF
10     CONTINUE

      RETURN
      END

C*****
C
C
      SUBROUTINE WHENFGT(N,ARRAY,INC,TARGET,INDEX,NVAL)
C
C
C*****
C
C
C      SUBROUTINE TO RETURN LOCATIONS OF ALL ELEMENTS OF A REAL
C      VECTOR GREATER THAN TARGET.
C
C
C=====

      DOUBLE PRECISION ARRAY(1),TARGET
      INTEGER INDEX(1),N,INC,NVAL,I

      NVAL=0

      DO 10 I=1,N,INC
        IF (ARRAY(I) .GT. TARGET) THEN
          NVAL=NVAL+1
          INDEX(NVAL)=I
        ENDIF
10     CONTINUE

      RETURN
      END

C*****
C
C
      SUBROUTINE WHENFLT(N,ARRAY,INC,TARGET,INDEX,NVAL)
C
C

```

```

C*****
C
C
C   SUBROUTINE TO RETURN LOCATIONS OF ALL ELEMENTS OF A REAL
C   VECTOR LESS THAN TARGET.
C
C
C=====

```

```

      DOUBLE PRECISION ARRAY(1), TARGET
      INTEGER INDEX(1), N, INC, NVAL, I

      NVAL=0

      DO 10 I=1, N, INC
        IF (ARRAY(I) .LT. TARGET) THEN
          NVAL=NVAL+1
          INDEX(NVAL)=I
        ENDIF
10    CONTINUE

      RETURN
      END

```

```

C*****
C
C
C   SUBROUTINE WHENIEQ(N, IARRAY, INC, ITARGET, INDEX, NVAL)
C
C
C*****
C
C
C   SUBROUTINE TO RETURN LOCATIONS OF ALL ELEMENTS OF AN INTEGER
C   VECTOR GREATER THAN ITARGET.
C
C
C=====

```

```

      INTEGER IARRAY(1), INDEX(1), N, INC, ITARGET, NVAL, I

      NVAL=0

      DO 10 I=1, N, INC
        IF (IARRAY(I) .EQ. ITARGET) THEN
          NVAL=NVAL+1
          INDEX(NVAL)=I
        ENDIF

```

```
10      CONTINUE
```

```
      RETURN
      END
```

```
C*****
C
C
```

```
      SUBROUTINE WHENIGE(N,IARRAY,INC,ITARGET,INDEX,NVAL)
```

```
C
C
C*****
```

```
C      SUBROUTINE TO RETURN LOCATIONS OF ALL ELEMENTS OF AN INTEGER
C      VECTOR GREATER THAN ITARGET.
```

```
C
C
C=====
```

```
      INTEGER IARRAY(1),INDEX(1),N,INC,ITARGET,NVAL,I
```

```
      NVAL=0
```

```
      DO 10 I=1,N,INC
        IF(IARRAY(I).GE.ITARGET) THEN
          NVAL=NVAL+1
          INDEX(NVAL)=I
        ENDIF
```

```
10      CONTINUE
```

```
      RETURN
      END
```

```
C*****
C
C
```

```
      SUBROUTINE WHENIGT(N,IARRAY,INC,ITARGET,INDEX,NVAL)
```

```
C
C
C*****
```

```
C      SUBROUTINE TO RETURN LOCATIONS OF ALL ELEMENTS OF AN INTEGER
C      VECTOR GREATER THAN ITARGET.
```

```
C
C
C=====
```

```

      INTEGER IARRAY(1), INDEX(1), N, INC, ITARGET, NVAL, I

      NVAL=0

      DO 10 I=1, N, INC
        IF (IARRAY(I) .GT. ITARGET) THEN
          NVAL=NVAL+1
          INDEX(NVAL)=I
        ENDIF
10     CONTINUE

      RETURN
      END

C*****
C
C
      SUBROUTINE WHENILE(N, IARRAY, INC, ITARGET, INDEX, NVAL)
C
C
C*****
C
C
C      SUBROUTINE TO RETURN LOCATIONS OF ALL ELEMENTS OF AN INTEGER
C      VECTOR LESS THAN OR EQUAL TO ITARGET.
C
C
C=====

      INTEGER IARRAY(1), INDEX(1), N, INC, ITARGET, NVAL, I

      NVAL=0

      DO 10 I=1, N, INC
        IF (IARRAY(I) .LE. ITARGET) THEN
          NVAL=NVAL+1
          INDEX(NVAL)=I
        ENDIF
10     CONTINUE

      RETURN
      END

C*****
C
C
      SUBROUTINE WHENILT(N, IARRAY, INC, ITARGET, INDEX, NVAL)
C

```

```

C
C*****
C
C
C      SUBROUTINE TO RETURN LOCATIONS OF ALL ELEMENTS OF AN INTEGER
C      VECTOR LESS THAN ITARGET.
C
C=====

      INTEGER IARRAY(1), INDEX(1), N, INC, ITARGET, NVAL, I

      NVAL=0

      DO 10 I=1,N,INC
        IF(IARRAY(I).LT.ITARGET) THEN
          NVAL=NVAL+1
          INDEX(NVAL)=I
        ENDIF
10    CONTINUE

      RETURN
      END

C*****
C
C
C      SUBROUTINE WHENNE(N, IARRAY, INC, ITARGET, INDEX, NVAL)
C
C
C*****
C
C
C      SUBROUTINE TO RETURN LOCATIONS OF ALL ELEMENTS OF AN INTEGER
C      VECTOR NOT EQUAL TO ITARGET.
C
C=====

      INTEGER IARRAY(1), INDEX(1), N, INC, ITARGET, NVAL, I

      NVAL=0

      DO 10 I=1,N,INC
        IF(IARRAY(I).NE.ITARGET) THEN
          NVAL=NVAL+1
          INDEX(NVAL)=I
        ENDIF

```

```

10      CONTINUE

        RETURN
      END
@PROCESS DC(BODYCEL,CONCOM,GAS)
C*****
C
      SUBROUTINE OUTDISP(N)
C
C-----
C
C      SUBROUTINE TO OUTPUT THE POSITIONS OF THE PARTICLES AT EVERY
C      QUARTER TURN (TD), AS WELL AS THE VELOCITY DISPERSION AND
C      AVERAGE ROTATIONAL AND RADIAL VELOCITIES AT ONE KPC INTERVALS.
C      THIS SUBROUTINE ALSO CALCULATES THE MASSFRACTION AND
C      NUMBER DENSITY AT ONE KPC INTERVALS.
C      R. DRIMMEL '91
C
C=====

/*INCLUDE NEWHEAD FORTRAN A

C
C      LOCAL VARIABLES
C
C-----

      DOUBLE PRECISION VT,VR,R,VRAV(50),VTAV(50),SR(50),ST(50),
&          SINA,COSA,MFRC,AREA,RAD,MASSP,DX,DY,RDEN,
&          AM(50),VX,VY,MDEN(50),SUMNR,SW(50)
      INTEGER I,J,P,NDIMO,NR(50),N,M

      PI=4.0D0*DATAN(1.0D0)
      NDIMO=NDIM
      DO J=1,50
        VTAV(J)=0.
        VRAV(J)=0.
        SR(J)=0.
        ST(J)=0.
        AM(J)=0.
        MDEN(J)=0.
        NR(J)=0
        SW(J)=0.
      ENDDO

      DO I=1+NMP,NGP-NAGP
        P=BODLIST(I)
        DX=POS(P,1)-CMPOS(1)

```

```

DY=POS(P,2)-CMPOS(2)
R=DSQRT(DX*DX+DY*DY)
M=INT(R/ANW)+1
IF(M.LE.50) THEN
  RDEN=R*RH(P)
  VX=VEL(P,1)-CMVEL(1)
  VY=VEL(P,2)-CMVEL(2)
  COSA=DX/R
  SINA=DY/R
  VT=VY*COSA - VX*SINA
  VR=VY*SINA + VX*COSA
  NR(M)=NR(M) + 1
  VRAV(M)=VRAV(M)+VR/RDEN
  VTAV(M)=VTAV(M)+VT/RDEN
  SR(M)=SR(M)+VR*VR/RDEN
  ST(M)=ST(M)+VT*VT/RDEN
  AM(M)=AM(M)+R*VT*MASS(P)
  MDEN(M)=MDEN(M)+MASS(P)
  SW(M)=SW(M)+1.D0/RDEN
ENDIF
ENDDO

SUMNR=0
WRITE(16,*) TNOW
DO J=1,50
  RAD=J*ANW
  IF(NR(J).NE.0) THEN
    VTAV(J)=VTAV(J)/SW(J)
    VRAV(J)=VRAV(J)/SW(J)
    SR(J)=SR(J)/SW(J) - VRAV(J)*VRAV(J)
    ST(J)=ST(J)/SW(J) - VTAV(J)*VTAV(J)
  ENDIF
  WRITE(16,10) VTAV(J),VRAV(J),ST(J),SR(J),(RAD-0.5*ANW),AM(J)
ENDDO

IF(N.NE.0) THEN
  WRITE(17,*) TNOW
  WRITE(17,20) 0.D0,0.D0,0.D0,LT(0),0
  LT(0)=0.D0
  DO J=1,50
    RAD=J*ANW
    AREA=PI*(2.0D0*RAD - ANW)*ANW
    SUMNR=SUMNR+MDEN(J)
    MFRC=SUMNR/MTOT
    MDEN(J)=MDEN(J)/AREA
    WRITE(17,20) MFRC,MDEN(J),RAD,LT(J),NR(J)
    LT(J)=0.
  ENDDO

```

```
ENDIF  
  
10  FORMAT(1X,D14.7,1X,3 (D12.5,1X) ,F7.3,1X,D14.8)  
20  FORMAT(1X,G12.5,2X,G16.9,2X,F7.3,2X,G16.9,2X,I5)  
  
RETURN  
END
```

```

C*****
C
C
C   THIS IS THE HEADER FOR ALL SUBROUTINES IN T2DSPH.
C
C
C=====
C   CHARACTER*50 HEADLIN
C   CHARACTER*3  SOFT
C   DOUBLE PRECISION MASS,TOL,TOL2INV,EPS,RSIZE,RBMIN,PHI,POS,VEL,
&       ACC,QUAD,TNOW,TPOS,DTIME,DTIME2,TINY,ONE,TWO,
&       CPUT0,CPUT1,CPUTIME,TD,TDUMP,ZERO,CELLSIZ,
&       MTOT,ETOT,EKTOT,EPTOT,UTOT,MINUST,CMPOS,CMVEL,
&       PHSMOOT,ACSMOOT,CN,PI,MASS1,LT,ANW,VET,AMVEC,
&       TELAPS,DTS,DT,DVEL,SOF,ZETA,QTOT,Q,RMIN,RMAX
C   INTEGER ROOT,SUBP,NBODIES,INCELLS,NTTOT,NTMIN,NTMAX,NTAVG,
&       NSTEPS,NOUTBOD,NOUTLOG,NDIM,NSUBCEL,NBODSMA,NCELLS,
&       NBODCEL,NBODS1,NBODLIS,NINTERP,VTYP,IDUM,MINDX,NGPTS,
&       NADVLIS,NNBSLIS,TSTEP,BNOW,NMAX,TBIN,NCMAX,NGAS
C   LOGICAL USEQUAD,REBOX

C   PARAMETER (NDIM=2,NSUBCEL=2**NDIM,NBINMAX=10)
C   PARAMETER (NBODSMA=11000,NCELLS=20000,NGPTS=50)
C   PARAMETER (NBODCEL=NBODSMA+NCELLS,NBODS1=NBODSMA+1)
C   PARAMETER (NINTERP=2000)

C   COMMON/PARAMCO/TOL,TOL2INV,EPS,CN,ANW,SOF,ZETA,NBODIES,NBODLIS,
&       NADVLIS,NGAS,NNBSLIS,REBOX,USEQUAD
C   COMMON/MSGCOM/HEADLIN,SOFT
C   COMMON/CELLCOM/RSIZE,RBMIN(NDIM),CMPOS(NDIM),CMVEL(NDIM),
&       AMVEC(3),INCELLS
C   COMMON/POINTER/ROOT,SUBP(NBODS1:NBODCEL,1:NSUBCEL)
C   COMMON/BODYCEL/MASS(1:NBODCEL),PHI(1:NBODSMA),Q(1:NBODSMA),
&       POS(0:NBODCEL,1:NDIM),CELLSIZ(1:NBODCEL),
&       VEL(1:NBODSMA,1:NDIM),ACC(1:NBODSMA,1:NDIM),
&       VET(1:NBODSMA,1:NDIM),DVEL(1:NBODSMA,1:NDIM),
&       DT(1:NBODSMA),TBIN(NBODSMA),MINDX(NBODSMA)
C   COMMON/QUADCOM/QUAD(NBODS1:NBODCEL,1:2*NDIM-1)
C   COMMON/FORCECO/NTTOT,NTMIN,NTMAX,NTAVG,NMAX,NCMAX
C   COMMON/TIMECOM/TD,TDUMP,TNOW,TPOS,DTIME,DTIME2,DTS,TELAPS,
&       RMIN(0:NBINMAX),RMAX(0:NBINMAX),
&       NSTEPS,NOUTBOD,NOUTLOG,TSTEP,BNOW
C   COMMON/CPUCOM/CPUT0,CPUT1,CPUTIME
C   COMMON/MISCCOM/TINY,ZERO,ONE,TWO,MINUST,PI,MASS1,IDUM
C   COMMON/ENRGYCO/MTOT,ETOT,EKTOT,EPTOT,UTOT,QTOT,LT(0:50)
C   COMMON/GRAVCOM/PHSMOOT(0:1+NINTERP),ACSMOOT(0:1+NINTERP)

C=====

```

C DEFINITIONS SPECIFIC TO INPUT/OUTPUT.

```
C=====
      INTEGER UTERM,UPARS,ULOG,UBODSIN,UBODSOU,UBODSAS
      CHARACTER*8 PARSFIL,LOGFILE,IBODFIL,OBODFIL,OASCFIL

      PARAMETER(UTERM=6,UPARS=10,ULOG=11,UBODSIN=12,UBODSOU=13,
&              UBODSAS=14)
      PARAMETER(PARSFIL='TREEPAR',LOGFILE='TREELOG',
&              IBODFIL='TREEBI',OBODFIL='TREEBO',
&              OASCFIL='TREEBOA')
```

C=====

C DEFINITIONS SPECIFIC TO VECTORIZED TREE CONSTRUCTION, VECTORIZED

C AND TREE WALK.

```
C=====
      DOUBLE PRECISION WORKVEC
      INTEGER SUBINDE,ACTLIST,BODLIST,TEMPLIS,PARENT,ASUBP,CELLIST,
&              SUBPVEC(NSUBCEL*NCELLS),ISUBSET,NWORKVE,ADVLIST,
&              NBSLIST
      PARAMETER(NWORKVE=15000)

      EQUIVALENCE (SUBPVEC(1),SUBP(NBODS1,1))

      COMMON/CONCOM/CELLIST(NCELLS),PARENT(NBODSMA),ASUBP(NBODSMA),
&              TEMPLIS(NBODSMA),ACTLIST(NBODSMA),BODLIST(NBODSMA),
&              NBSLIST(NBODSMA),ADVLIST(NBODSMA),
&              ISUBSET(NBODSMA),SUBINDE(NBODSMA)
      COMMON/WORKCOM/WORKVEC(NWORKVE)
```

C

C=====

C DEFINITIONS SPECIFIC TO GASEOUS PARTICLES

```
C=====
      INTEGER NGPMAX,NNBMAX,NSTMAX,NNBS,LIST,NGP,SLIST,INDL,NNB
      PARAMETER(NGPMAX=11000,NNBMAX=24,NSTMAX=200)
      DOUBLE PRECISION RH,VS2,GAMMA,H,VW3,VDIVW3,ALFA,BHTA,
&              C1,C2,HMAX,GMONE,GMTWO,C,F,DELV,RHS,DRHDT,
&              NU,AVM,ISS,GRADV,UDOT,U
      LOGICAL SWITCH
      COMMON/GASPAR/VS2,GAMMA,GMONE,GMTWO,ALFA,BHTA,NU,C1,C2,ISS,
&              HMAX,NGP,SWITCH
      COMMON/GAS/ RH(NGPMAX),VW3(2+NINTERP),VDIVW3(2+NINTERP),
&              RHS(NGPMAX),DRHDT(NGPMAX),AVM(NGPMAX),H(NGPMAX),
&              C(NGPMAX),F(NGPMAX),DELV(NGPMAX),U(NGPMAX),
&              GRADV(NGPMAX,4),UDOT(NGPMAX),
&              SLIST(5*NNBMAX/2,NGPMAX),INDL(NGPMAX),NNB(NGPMAX),
&              LIST(2*NNBMAX,NGPMAX),NNBS(NGPMAX)
```

C=====

C

C DEFINITIONS SPECIFIC TO ACCRETION

C

C=====

DOUBLE PRECISION TAM, TAAM, TAEK, TAE, ARAD, MDOT

INTEGER NMP, NAGP, NACC, ACCLIST, NCP

COMMON/ACCPAR/ TAM, TAAM(3), TAEK, TAE, ARAD,

& MDOT, NACC, NAGP, NCP, NMP, ACCLIST(1000)

APPENDIX B
Algorithm for T2DSPH

The table on the following page contains the list of subroutines called by the main program, or by three main subroutines listed at the top of the table.

MAIN	INITSYS	STEPSYS	HALFSTP
INITSYS	SECONDS	If(telapse \neq 0)	CORVEL
If(N<NSTEPS)	STARTOU	HALFSTEP	ZEROACC
STEPSYS	INPARAM	else	GRAVITY
If(N \neq NOLD)	INBODS	ESTVEL	MAKESAP
OUTSTAT	INITPAR	STEPPOS	MAKELIS
ENDDO	INITSOE	endif	DENSITY
ENDRUN	ZEROACC	Build ADVLIST	HYDRO
	ZEROPOT	HALFSTP	HADJUST
	ACCRET(0)	AMFLUX	STEPVEL
	Initialize:	ACCRET	STEPPOS
	ADVLIST	COURANT(N)	ESTVEL
	NBSLIST	If(telapse=0)	
	GRAVITY	GRIDEN	
	INITLIS		
	DENSITY		
	HYDRO		
	COURANT(N)		
	GRIDEN		
	OUTSTAT(0)		
	Do first step:		
	STEPSYS		

APPENDIX C

PROGRAM EXPD

```

PROGRAM EXPD

C
C   This program builds an exponential disk from concentric
C   rings. The disk is built from the center outwards by
C   making the distance between particles in a ring approx.
C   equal to the distance from the previous ring. This version
C   adds a perturbation to the density by displacing the particle
C   positions, and care is taken to remove the perturbation in
C   the velocities.
C
C=====
      IMPLICIT NONE
      INTEGER I,J,NR,NRMAX,NUM,NBODMAX,NINTERP,SMINDX,FLAG2,INRNG,
&    NACP,IDUM
      PARAMETER(NRMAX=300,NBODMAX=25000,NINTERP=2000)
      INTEGER N(NRMAX),NTOT,FLAG,NRAD,ERROR,NRNG,NP,PNP,NMP
      DOUBLE PRECISION R(NRMAX),THETA,DTH,X0,Y0,PI,DX,DY,GAMMA,
&    MD,RD,MC,RS,MASSP,RP,RAD,N0,DS,ADS,LADS,PRP,EPS,VDIVR,DR,
&    TINY,VX(NBODMAX),VY(NBODMAX),V02(NRMAX),ACC(NBODMAX,2),
&    X(NBODMAX),Y(NBODMAX),H(NBODMAX),MASS(NBODMAX),DEN(NRMAX),
&    V2(NRMAX),MTOT,QMIN,VRDR,VR,VT,U(NBODMAX),GMONE,DENJ,HP,
&    ACSMOOT(0:NINTERP+1),XW,XW2,XW3,XW4,DELDRG,DEN0,ISS,ARAD,
&    DR1,VT2,VT1,R2,R1,VT0,R0,DR0,DR2,DX0,DX1,DX2,ALFA,ALFP,AMP

C
C   VARIABLES FOR SELF GRAVITY ONLY:
C-----
      DOUBLE PRECISION MASS1,ACCSM,DACCX,R3INV,DACCY,RNIN,ACCX,ACCY,
&    DRSM,RINV

C
C   VARIABLES FOR PRESSURE SUPPORT:
C-----
      DOUBLE PRECISION Q(NRMAX),K2DMIN,KAPPA2(NRMAX),DV2DR,MINQ

      CHARACTER*1 QUERY
      COMMON/DISK/ RD,RS,N0,PI,NTOT

      OPEN(UNIT=10,FILE='expd.out',STATUS='UNKNOWN')
      OPEN(UNIT=20,FILE='expvel.dat',STATUS='UNKNOWN')
      OPEN(UNIT=30,FILE='expq.dat',STATUS='UNKNOWN')
      OPEN(UNIT=40,FILE='exppar.dat',STATUS='UNKNOWN')

```

```

WRITE(6,*) 'INPUT THE MASS OF THE DISK:'
READ(5,*) MD
WRITE(6,*) ' '
WRITE(6,*) 'INPUT THE MASS OF THE CENTRAL OBJECT:'
READ(5,*) MC
WRITE(6,*) ' '
C   WRITE(6,*) 'INPUT THE SCALE LENGTH:'
C   READ(5,*) RS
C   WRITE(6,*) ' '
C   WRITE(6,*) 'INPUT THE RADIUS OF THE DISK:'
C   READ(5,*) RD
C   WRITE(6,*) ' '
WRITE(6,*) 'INPUT THE RATIO OF SPECIFIC HEATS.'
READ(5,*) GAMMA

IDUM=-1
MTOT=MD+MC
RD=1.D0
RS=0.25D0
NMP=1
NACP=0
GMONE=GAMMA-1.D0
ERROR=0
ALFA=0.D0

C   WRITE(6,*) 'WHAT IS THE VISCOUS PARAMETER ALFA?'
C   READ(5,*) ALFA
WRITE(6,*) 'INPUT THE PERTURBATION AMPLITUDE.'
READ(5,*) AMP
WRITE(6,*) 'AT WHAT RADIUS DO YOU WISH TO HAVE ARAD?'
READ(5,*) ARAD
WRITE(6,*) ' '
WRITE(6,*) 'WHAT IS THE DESIRED GRAVITATIONAL SMOOTHING LENGTH?'
READ(5,*) EPS
WRITE(6,*) ' '
WRITE(6,*) 'DO YOU WISH TO INPUT THE MINIMUM Q VALUE? (Y/N)'
READ(5,*) QUERY
IF(QUERY.EQ.'Y'.OR.QUERY.EQ.'y') THEN
    WRITE(6,*) ' '
    WRITE(6,*) 'INPUT QMIN:'
    READ(5,*) QMIN
    QUERY='Y'
ELSE
    WRITE(6,*) ' '
    WRITE(6,*) 'INPUT THE ISOTHERMAL SOUND SPEED:'
    READ(5,*) ISS
    QUERY='N'
ENDIF

```

```

WRITE(40,*) '***** EXPONENTIAL DISK *****'
WRITE(40,*) ' '
WRITE(40,*) ' '
WRITE(40,100) MC,MD,RD,RS
WRITE(40,*) ' '
100 FORMAT(1X,' MC = ',D12.5,3X,'MD= ',D12.5,3X,'RD= ',D12.5,
& 3X,'RS= ',D12.5)
200 FORMAT(1X,' GAMMA= ',D12.4,3X,'QMIN= ',D12.5,3X,'EPS= ',D12.5)
300 FORMAT(1X,' GAMMA= ',D12.4,3X,'ISS= ',D12.5,3X,'EPS= ',D12.5)
C
C INITIALIZE VARIABLES AND SMOOTHED ACCELERATION TABLE
C-----
DELD RG=2./NINTERP
DO I=0,NINTERP/2
  XW=I*DELD RG
  XW2=XW*XW
  XW3=XW2*XW
  ACSMOOT(I)=XW3*(4./3.-6.*XW2/5.+0.5*XW3)
ENDDO
DO I=NINTERP/2+1,NINTERP
  XW=I*DELD RG
  XW2=XW*XW
  XW3=XW2*XW
  XW4=XW2*XW2
  ACSMOOT(I)=-1./15.+8.*XW3/3.-3.*XW4+6.*XW3*XW2/5.-XW4*XW2/6.
ENDDO
ACSMOOT(I)=1.D0

WRITE(6,*) 'INPUT THE TOTAL NUMBER OF PARTICLES YOU DESIRE:'
READ(5,*) NTOT

PI=4.D0*DATAN(1.D0)
TINY=1.D-20
DO I=1,NRMAX
  N(I)=0
  R(I)=0.D0
  V02(I)=0.D0
ENDDO
DO I=1,NBODMAX
  ACC(I,1)=0.D0
  ACC(I,2)=0.D0
ENDDO
C
C FIND THE RADIUS OF THE INNER DISK
C-----
FLAG2=0
15 NR=7-NMP
THETA=0.D0

```

```

N0=DFLOAT(NTOT) / (2.D0*PI*RS*RS*(1.D0-DEXP(-RD/RS)*(RD/RS+1.D0)))
DEN0=N0*MD/DFLOAT(NTOT)
RAD=0.01D0
CALL FINDR(NR,RAD,ERROR)
IF(ERROR.NE.0) THEN
    WRITE(6,*) 'ERROR',I
    GOTO 99
ENDIF
IF(NMP.EQ.1) THEN
    I=1
    N(I)=6
    R(I)=DSQRT(0.5D0)*RAD
ELSE
    N(1)=1
    R(1)=0.D0
    N(2)=6
    R(I)=DSQRT(5.D0/8.D0)*RAD
ENDIF

```

C

C BUILD REMAINING DISK

C-----

```

DOWHILE(NR.LT.NTOT.AND.I.LT.199)
    I=I+1
    NP=N(I-1)*8/10
    NRAD=NR+NP
    CALL FINDR(NRAD,RAD,ERROR)
    IF(ERROR.NE.0) THEN
        WRITE(6,*) 'ERROR'
        GOTO 99
    ENDIF
    RP=DSQRT(0.5D0*(RAD**2+R(I-1)**2))
    DS=RP-R(I-1)-2.D0*PI*RP/DFLOAT(NP)
    ADS=DABS(DS)
    NP=NP+1
    LADS=ADS+1.D0
    FLAG=0
    DOWHILE(LADS.GT.ADS.AND.FLAG.LT.100)
        FLAG=FLAG+1
        LADS=ADS
        PRP=RP
        PNP=NP
        NRAD=NR+NP
        CALL FINDR(NRAD,RAD,ERROR)
        IF(ERROR.NE.0) THEN
            WRITE(6,*) 'ERROR'
            GOTO 99
        ENDIF
    ENDIF

```

```

ENDIF
RP=DSQRT(0.5D0*(RAD**2+R(I-1)**2))
DS=RP-R(I-1)-2.D0*PI*RP/DFLOAT(NP)
IF(DS.LT.0.D0) THEN
    NP=NP+1
ELSE
    NP=NP-1
ENDIF
ADS=DABS(DS)
ENDDO
N(I)=PNP
R(I)=PRP
NR=NR+N(I)
ENDDO
WRITE(6,*) 'NR=',NR
IF(FLAG2.GT.20) THEN
    WRITE(6,*) 'CANNOT FIND INTEGER NUMBER OF PARTICLES FOR DISK.'
    GOTO 99
ENDIF
IF(NR.NE.NTOT) THEN
    NTOT=NR
    FLAG2=FLAG2+1
    GOTO 15
ENDIF
NRNG=I
NTOT=NR+NMP
IF(I.GT.199) THEN
    WRITE(6,*) 'NUMBER OF RINGS IS GREATER THAN 199'
    WRITE(6,*) FLAG,N(I-1),N(I),R(I)
    GOTO 25
ENDIF
WRITE(6,*) 'NTOT=',NTOT

DO I=1,NRNG
    DEN(I)=DEN0*DEXP(-R(I)/RS)
    IF(R(I).GT.ARAD.AND.R(I-1).LT.ARAD) THEN
        ARAD=(R(I)+R(I-1))*0.5D0
        INRNG=I
    ENDIF
ENDDO

C    EPS=ARAD*0.5D0
    IF(QUERY.EQ.'Y') THEN
        WRITE(40,200) GAMMA,QMIN,EPS
    ELSE
        WRITE(40,300) GAMMA,ISS,EPS
    ENDIF
    WRITE(40,*) ' '

```

```

WRITE(40,*) '      ALFA=',ALFA,'      ARAD=',ARAD
WRITE(40,*) '      '
WRITE(40,*) '      NTOT=',NTOT,'      NRNG=',NRNG
WRITE(40,*) '      '
WRITE(40,*) '      NO=',NO,'      DEN0=',DEN0
WRITE(40,*) '      '

C
C  CALCULATE X AND Y COORDINATE OF EACH PARTICLE
C-----
      MASSP=MD/DFLOAT(NR)
      X(1)=0.D0
      Y(1)=0.D0
      H(1)=ARAD
      MASS(1)=MC
      J=2-NMP
      RAD=R(J)
      HP=3.D0*RAD
      NUM=1+N(J)
      DTH=2.D0*PI/DFLOAT(N(J))
      DENJ=DEN(J)
      DO I=2,NUM
        X(I)=RAD*DCOS(THETA)
        Y(I)=RAD*DSIN(THETA)
        H(I)=HP
        U(I)=DENJ**GMONE/GMONE
        MASS(I)=MASSP
        THETA=THETA+DTH
      ENDDO
      DO J=3-NMP,NRNG
        THETA=THETA+DTH*0.5D0
        DTH=2.D0*PI/DFLOAT(N(J))
        RAD=R(J)
        HP=3.D0*(RAD-R(J-1))
        DENJ=DEN(J)
        DO I=1,N(J)
          NUM=NUM+1
          X(NUM)=RAD*DCOS(THETA)
          Y(NUM)=RAD*DSIN(THETA)
          MASS(NUM)=MASSP
          H(NUM)=HP
          U(NUM)=DENJ**GMONE/GMONE
          THETA=THETA+DTH
        ENDDO
      ENDDO
      IF(NUM.NE.NTOT) THEN
        WRITE(6,*) 'NUM.NE.NR'
        GOTO 99
      ENDIF

```

```

C
C  FIND MASS WITHIN ARAD
C-----
      MASS1=0.D0
      IF (NMP.EQ.1) MASS1=MASS(NMP)
      NUM=NMP
      DO J=1, INRNG-1
        DO I=1, N(J)
          NUM=NUM+1
          MASS1=MASS1+MASS(NUM)
          MASS(NUM)=0.0
        ENDDO
      ENDDO
      NACP=NUM

C
C  SUM OVER ALL PARTICLES FOR EACH PARTICLE TO FIND ACCELERATION
C-----
C      TEMPORARILY SET MASS(1)=MASS1
C-----
      MASS(1)=MASS1
      DO J=1, NTOT-1
        X0=X(J)
        Y0=Y(J)
        ACCX=0.D0
        ACCY=0.D0
        DO I=J+1, NTOT
          DX=X(I)-X0
          DY=Y(I)-Y0
          RAD=DSQRT(DX**2+DY**2)
          RINV=1.D0/(RAD+TINY)
          R3INV=RINV/(RAD*RAD+TINY)
          RNIN=RAD*NINTERP/(2.D0*EPS)
          SMINDX=RNIN
          SMINDX=MIN(NINTERP, SMINDX)
          DRSM=MIN(1, RNIN-SMINDX)
          ACCSM=(1.-DRSM)*ACSMOOT(SMINDX)+DRSM*ACSMOOT(1+SMINDX)
          R3INV=ACCSM*R3INV
          DACCX=R3INV*DX
          DACCY=R3INV*DY
          ACCX=ACCX+DACCX
          ACCY=ACCY+DACCY
          ACC(I,1)=ACC(I,1)-DACCX*MASS(J)
          ACC(I,2)=ACC(I,2)-DACCY*MASS(J)
        ENDDO
        ACC(J,1)=ACC(J,1)+ACCX*MASSP
        ACC(J,2)=ACC(J,2)+ACCY*MASSP
      ENDDO
C

```

C THE FOLLOWING IS FOR A MASSLESS DISK WITH NO PRESSURE SUPPORT.

C-----

```

C      NUM=NACP+NMP
C      R(NRNG+1)=R(NRNG)+(R(NRNG)-R(NRNG-1))
C      DO J=INRNG,NRNG
C          DR=(R(J+1)-R(J-1))*0.1D0
C          DO I=1,N(J)
C              NUM=NUM+1
C              CALL GASDEV(DX,DY,IDUM)
C              X(NUM)=X(NUM)+DX*DR
C              Y(NUM)=Y(NUM)+DY*DR
C          ENDDO
C      ENDDO
C      X0=X(1)
C      Y0=Y(1)
C      VX(1)=0.D0
C      VY(1)=0.D0
C      ALFP=0.07D0*ALFA*9.D0*DSQRT(GAMMA*ISS*MASSP)/5.D0
C      DO J=2,NTOT
C          DX=X(J)-X0
C          DY=Y(J)-Y0
C          RAD=DSQRT(DX**2+DY**2)
C          VRDR=-ALFP*(1.D0-GAMMA*RAD/RS)/RAD**2
C          VDIVR=DSQRT(MASS(1)/RAD)/RAD
C          VX(J)=-Y(J)*VDIVR+X(J)*VRDR
C          VY(J)=X(J)*VDIVR+Y(J)*VRDR
C      ENDDO
C
C      RESET MASSES
C-----

```

```

C      NUM=NMP
C      DO J=1,INRNG-1
C          DO I=1,N(J)
C              NUM=NUM+1
C              MASS(NUM)=MASSP
C          ENDDO
C      ENDDO
C      MASS(1)=MC
C
C      CALCULATE THE VELOCITY CURVE
C-----

```

```

C      NUM=NMP
C      DO J=1,NRNG
C          V02(J)=0.D0
C          DO I=1,N(J)
C              NUM=NUM+1
C              V02(J)=V02(J)+ACC(NUM,1)*X(NUM)+ACC(NUM,2)*Y(NUM)

```

```

ENDDO
V02(J)=-V02(J)/DFLOAT(N(J))
IF(V02(J).LT.0.D0) THEN
  WRITE(6,*) 'V SQUARED < 0. FOR RING',J
  GOTO 99
ENDIF
ENDDO

```

```

C
C  FIND THE EPICYCLIC FREQUENCY AND ISOTHERMAL SOUND SPEED
C-----

```

```

K2DMIN=1.D06
DO J=2,NRNG-1
  DV2DR=(V02(J+1)-V02(J-1))/(R(J+1)-R(J-1))
  KAPPA2(J)=DV2DR/R(J)+2.D0*V02(J)/R(J)**2
  K2DMIN=MIN(K2DMIN,KAPPA2(J)*DEN(J)**(GAMMA-3.D0))
ENDDO
RAD=(V02(J)-V02(J-2))/(R(J)-R(J-2))
DV2DR=DV2DR+(R(NRNG)-R(J))*(DV2DR-RAD)/(R(J)-R(J-1))
KAPPA2(NRNG)=DV2DR/R(NRNG)+2.D0*V02(NRNG)/R(NRNG)**2
K2DMIN=MIN(K2DMIN,KAPPA2(J)/DEN(J)**(3.D0-GAMMA))
WRITE(6,*) 'MIN(KAPPA^2*DEN^(GAMMA-3))=',K2DMIN
IF(QUERY.EQ.'Y') THEN
  ISS=(QMIN*PI)**2/(GAMMA*K2DMIN)
  WRITE(6,*) 'THE ISOTHERMAL SOUND SPEED IS ',ISS
  WRITE(40,*) 'ISS=',ISS
ENDIF
MINQ=1.D06
DO J=2,NRNG-1
  Q(J)=DSQRT(KAPPA2(J)*GAMMA*ISS)/(PI*DEN(J)**((3.-GAMMA)/2.))
  MINQ=MIN(MINQ,Q(J))
  WRITE(30,*) Q(J),DSQRT(KAPPA2(J)),R(J)
ENDDO
WRITE(6,*) 'MINIMUM Q=',MINQ
WRITE(40,*) 'MINQ=',MINQ

```

```

C
C  FIND THE ROTATIONAL VELOCITY
C-----
ALFP=0.1D0*ALFA*9.D0*DSQRT(GAMMA*ISS*MASSP)/5.D0
DO J=1,NRNG
  V2(J)=-GAMMA*ISS*DEN(J)**GMONE*R(J)/RS+V02(J)
  IF(V2(J).LT.0.D0) THEN
    WRITE(6,*) 'V2 IS LT 0. V2=',V2(J),J
    GOTO 99
  ENDIF
  VR=-ALFP*(1.D0-GAMMA*R(J)/RS)/R(J)

```

```

      WRITE(20,*) R(J),DSQRT(V02(J)),DSQRT(V2(J)),VR
      ENDDO

```

```

C
C  ADD DENSITY PERTURBATION.
C-----

```

```

      NUM=NACP+NMP
      R(NRNG+1)=R(NRNG)+(R(NRNG)-R(NRNG-1))
      DO J=INRNG,NRNG
        DR=(R(J+1)-R(J-1))*AMP
        DO I=1,N(J)
          NUM=NUM+1
          CALL GASDEV(DY,DX,IDUM)
          X(NUM)=X(NUM)+DX*DR
          Y(NUM)=Y(NUM)+DY*DR
        ENDDO
      ENDDO

```

```

C
C  ASSIGN VELOCITIES TO PARTICLES
C-----

```

```

      NUM=NMP
      DO J=1,INRNG-1
        VDIVR=DSQRT(V2(J))/R(J)
        VRDR=-ALFP*(1.D0-GAMMA*R(J)/RS)/R(J)**2
        DO I=1,N(J)
          NUM=NUM+1
          VX(NUM)=-Y(NUM)*VDIVR+X(NUM)*VRDR
          VY(NUM)=X(NUM)*VDIVR+Y(NUM)*VRDR
        ENDDO
      ENDDO
      DR1=R(INRNG)-R(INRNG-1)
      VT2=DSQRT(V2(INRNG))
      VT1=DSQRT(V2(INRNG-1))
      R2=R(INRNG)
      R1=R(INRNG-1)
      DO J=INRNG,NRNG-1
        VT0=VT1
        VT1=VT2
        VT2=DSQRT(V2(J+1))
        R0=R1
        R1=R2
        R2=R(J+1)
        DR0=DR1
        DR1=R2-R1
        DR2=DR0+DR1
        DO I=1,N(J)
          NUM=NUM+1
          RAD=DSQRT(X(NUM)**2+Y(NUM)**2)

```

```

      VRDR=-ALFP*(1.D0-GAMMA*RAD/RS)/RAD**2
      DX0=RAD-R0
      DX1=RAD-R1
      DX2=RAD-R2
      VT=DX1*DX2*VT0/(DR0*DR2)-DX0*DX2*VT1/(DR0*DR1)+
&      DX0*DX1*VT2/(DR2*DR1)
      VX(NUM)=-Y(NUM)*VT/RAD+X(NUM)*VRDR
      VY(NUM)=X(NUM)*VT/RAD+Y(NUM)*VRDR
      ENDDO
    ENDDO
    J=NRNG
    DO I=1,N(J)
      NUM=NUM+1
      RAD=DSQRT(X(NUM)**2+Y(NUM)**2)
      VRDR=-ALFP*(1.D0-GAMMA*RAD/RS)/RAD**2
      DX0=RAD-R0
      DX1=RAD-R1
      DX2=RAD-R2
      VT=DX1*DX2*VT0/(DR0*DR2)-DX0*DX2*VT1/(DR0*DR1)+
&      DX0*DX1*VT2/(DR2*DR1)
      VX(NUM)=-Y(NUM)*VT/RAD+X(NUM)*VRDR
      VY(NUM)=X(NUM)*VT/RAD+Y(NUM)*VRDR
    ENDDO
  C
  C  OUTPUT DISK
  C-----
25  WRITE(10,*) NTOT,2,0.D0,NTOT,1,0.D0
    WRITE(10,*) X(1),Y(1)
    WRITE(10,*) .0D0,0.D0
    WRITE(10,*) MC,0.D0,H(1)
    DO I=2,NTOT
      WRITE(10,*) X(I),Y(I)
      WRITE(10,*) VX(I),VY(I)
      WRITE(10,*) MASSP,ISS*U(I),H(I)
    ENDDO

99  STOP
    END

C*****
C
C  SUBROUTINE FINDR(N,R,ERROR)
C
C  Subroutine to find the radius R within which N particles reside.
C  in an exponential disk with scale length RS, radius RD, and NTOT
C  total particles in an exponential disk.
C
C=====

```

```

IMPLICIT NONE
INTEGER N, FLAG, NTOT, ERROR
DOUBLE PRECISION R, ROLD, F, DF, PI, NO, RD, RS, PI
COMMON/DISK/ RD, RS, NO, PI, NTOT

```

```

ROLD=2.D0*RD
FLAG=0
DOWHILE (DABS (R-ROLD) .GT. 1.D-08 .AND. FLAG.LT. 20)
    FLAG=FLAG+1
    F=2.D0*PI*NO*RS*RS*(1.D0-DEXP(-R/RS)*(R/RS+1.D0))-N
    DF=2.D0*PI*NO*R*DEXP(-R/RS)
    ROLD=R
    R=ROLD-F/DF
ENDDO
IF (FLAG.GE. 20) ERROR=1

RETURN
END

```

```

C=====
C
C   THIS HAS THE DEFINED FUNCTIONS NEEDED TO GENERATE A
C   GAUSSIAN DEVIATE, USING THE SUBROUTINE GASDEV.
C
C=====

```

```

*****

```

```

C
C   FUNCTION RAN3 (IDUM)
C
C-----
C
C   Returns a uniform random deviate between 0.0 and 1.0. Set IDUM
C   to any negative value to initialize or reinitialize the sequence.
C   If running on machine that is weak on integer arithmetic, then use
C   commented lines. Any other large value of MBIG and MSEED will
C   work. This routine is originally from Knuth, but the current
C   version is from Numerical Recipes by Press et al.
C
C=====

```

```

IMPLICIT NONE
DOUBLE PRECISION FAC, RAN3
INTEGER MBIG, MSEED, IFF, IDUM, MJ, MA(55), MK, II, MZ, K, I,
& INEXT, INEXTP
PARAMETER (MBIG=1000000000, MSEED=161803398, MZ=0, FAC=1./MBIG)
C   IMPLICIT DOUBLE PRECISION (M)
C   PARAMETER (MBIG=4000000., MSEED=1618033., MZ=0., FAC=1./MBIG)
DATA IFF/0/

```

```

IF (IDUM.LT.0.OR.IFF.EQ.0) THEN
  IFF=1
  MJ=MSEED-IABS(IDUM)
  MJ=MOD(MJ,MBIG)
  MA(55)=MJ
  MK=1
  DO I=1,64
    II=MOD(21*I,55)
    MA(II)=MK
    MK=MJ-MK
    IF (MK.LT.MZ) MK=MK+MBIG
    MJ=MA(II)
  ENDDO
  DO K=1,4
    DO I=1,55
      MA(I)=MA(I)-MA(1+MOD(I+30,55))
      IF (MA(I).LT.MZ) MA(I)=MA(I)+MBIG
    ENDDO
  ENDDO
  INEXT=0
  INEXTP=31
  IDUM=1
ENDIF
INEXT=INEXT+1
IF (INEXT.EQ.56) INEXT=1
INEXTP=INEXTP+1
IF (INEXTP.EQ.56) INEXTP=1
MJ=MA(INEXT)-MA(INEXTP)
IF (MJ.LT.MZ) MJ=MJ+MBIG
MA(INEXT)=MJ
RAN3=MJ*FAC
RETURN
END

```

C

SUBROUTINE GASDEV(VR,VT,IRAN)

C

C-----

C

C This subroutine generates velocity deviates for a particle with a
C normal distribution and unit variance.

C

C=====

IMPLICIT NONE

INTEGER IRAN

DOUBLE PRECISION V1,V2,R,FAC,VR,VT,RAN3

R=1.

```
DOWHILE(R.GE.1.)  
    V1=2.*RAN3(IRAN)-1.  
    V2=2.*RAN3(IRAN)-1.  
    R=V1*V1 + V2*V2  
ENDDO  
FAC = DSQRT(-2.*DLOG(R)/R)  
VR=V1*FAC  
VT=V2*FAC  
  
RETURN  
END
```

BIBLIOGRAPHY

- Abramowicz, M.A., Lanza, A., Spiegel, E.A., & Szuszkiewicz, E. 1992, *Nature*, **356**, 41
- Adams, F.C., Lada, C.J., & Shu, F.H. 1988, *Ap. J.*, **326**, 865
- Anzer, U., Borner, G., & Monaghan, J.J. 1987, *A&A*, **176**, 235
- Artymowicz, P. 1993, *Ap. J.*, **419**, 166
- Balbus, S.A., & Hawley, J.F. 1991, *Ap. J.*, **376**, 214
- Benz, W. 1990, *The Numerical Modelling of Nonlinear Stellar Pulsations*, ed. J.R. Buchler (Kluwer, Dordrecht, Netherlands), p. 269
- Binney, J., & Tremaine, S. 1987, *Galactic Dynamics*, (Princeton University Press, Princeton, New Jersey)
- Bodenheimer, P., York, H.W., Rozyczka, M., & Tohline, J.E. 1990, *Ap. J.*, **355**, 651
- Bonnell, I. 1994, *Smooth Particle Hydrodynamics in Astrophysics; Memorie S. A. It.*, **65**, 1101
- Boss, A.P. 1989, *P.A.S.P.*, **101**, 767
- Boss, A.P. 1992 in *Astrophysical Disks*, *Ann. NY Acad. Sci.*, **675**, 303
- Bouvier, J., Malbet, F. & Monin, J. 1994, *Astron. Space Sci.*, **212**, 159
- Cassen, P.M., Smith, B.F., Reynolds, R.T., Miller, R.H. 1981, *Icarus*, **48**, 377
- Clark, C.J. & Pringle, J.E. 1993, *M.N.R.A.S.*, **261**, 190

- Edwards, S., Strom, S.E., Hartigan, P., Strom, K.M., Hillenbrand, L.A., Herbst, W., Attridge, J., Merrill, K.M., Probst, R., & Gatley, I. 1993, *Astron. J.*, **106**, 372
- Flebbe, O. 1994, *Ap. J.*, **431**, 754
- Gingold, R., & Monaghan, J. J. 1977, *M.N.R.A.S.*, **181**, 375
- Hartmann, L., Kenyon, S., & Calvet, N. 1993, *Ap. J.*, **407**, 219
- Hawley, J.F., & Balbus, S.A. 1991, *Ap. J.*, **376**, 223
- Hayashi, C., Nakazawa, K. & Adachi, I. 1977, *Publ. Astron. Soc. Japan*, **29**, 163
- Heller C.H., 1993, *Ap. J.*, **408**, 337
- Herant, M. 1994, *Smooth Particle Hydrodynamics in Astrophysics; Memorie S.A.It.*, **65**, 1013
- Hernquist, L. 1987, *Ap. J Suppl.*, **64**, 715
- Hernquist, L. & Katz, N. 1989, *Ap. J. Suppl.*, **70**, 419
- Hillenbrand, L.A., Strom, S.E., Vrba, F.J., & Keene, J. 1992, *Ap. J.*, **397**, 613
- Hunter, J.H., & Schweiker, K.S. 1981, *Ap J.*, **243**, 1030
- Hunter, J.H., & Horak, T. 1983, *Ap. J.*, **265**, 402
- Koerner, D.W., Sargent, A.I., & Beckwith, S.V.W. 1993, *Icarus*, **106**, 2
- Laughlin, G. & Bodenheimer, P. 1994, *Ap. J.*, **436**, 335
- Lay, O., Carlstrom, J., Hills, R., & Phillips, T. 1994, *Ap. J. Let.*, **434**, L75
- Lin, D.N.C., & Pringle, J.E. 1987, *M.N.R.A.S.*, **225**, 607
- Lin, D.N.C., & Pringle, J.E. 1990, *Ap. J.*, **356**, 515

- Lucy, L.B. 1977, *Ast. J.*, **82**, 1013
- Lynden-Bell, D., & Pringle, J.E. 1974, *M.N.R.A.S.*, **168**, 603
- Monaghan, J.J. 1992, *A.R.A.A.*, **30**, 543
- Narayan, R., Loeb, A., & Kumar, P. 1994, *Ap. J.*, **431**, 359
- Papaloizou, J.C.B., & Pringle, J.E. 1984, *M.N.R.A.S.*, **208**, 721
- Papaloizou, J.C.B., & Pringle, J.E. 1985, *M.N.R.A.S.*, **213**, 799
- Papaloizou, J.C. & Savonije, G.J. 1991, *M.N.R.A.S.*, **248**, 353
- Press, W., Teukolsky, S., Vetterling, W., & Flannery, B. 1992, *Numerical Recipes*, (Cambridge University Press, Cambridge, UK)
- Pringle, J.E. 1981, *A.R.A.A.*, **19**, 137
- Rydgren, A.E., & Zak, D.S. 1987, *P.A.S.P.*, **99**, 141
- Savonije, G.J., Papaloizou, J.C., & Heemskerk, M.H.M. 1992, in *Astrophysical Disks; Ann. N. Y. Acad. Sci.*, **675**, 264
- Shakura, N.I., & Sunyaev, R.A. 1973, *A&A*, **24**, 337
- Shu, F.H., Adams, F.C., & Lizano, S. 1987, *A.R.A.A.*, **25**, 23
- Steinmetz, M., & Muller, E. 1993, *A.&A.*, **268** 391
- Tohline, J.E., & Woodward, J.W. 1992 in *Astrophysical Disks; Ann. N.Y. Acad. Sci.*, **675**, 31
- Tomley, L., Cassen, P. & Steinman-Cameron, T. 1991, *Ap. J.*, **382**, 530
- Toomre, A. 1964, *Ap. J.*, **139**, 1217
- Wetherill, G. 1990, *Ann. Rev. Earth Planet Sci.*, **18**, 205

Wetherill, G. 1994, in *Planetary Systems: Formation, Evolution, and Detection*, eds. Burke, B., Rahe, J., Roettger, E. (Kluwer, Dordrecht, Netherlands)

Whitney, B., & Hartmann, L. 1993, *Ap. J.*, **402**, 605

York, H.W., Bodenheimer, P., & Laughlin, G. 1993, *Ap.J.*, **411**, 274

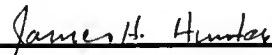
York, H.W., Bodenheimer, P., & Laughlin, G. 1995, *Ap.J.*, **443**, 199

Zurek, W.H., & Benz, W. 1986, **308**, 123


BIOGRAPHICAL SKETCH

Ronald Drimmel was born and baptized in Omaha, Nebraska, in 1964, events which he has no memory of. His first memories are of living in Osawatomi, Kansas, though he soon moved on, with his parents, to Wyoming. There his introverted self found solace in the surrounding hills of Rawlins, and the concept of space became attached to his dreams. Later, while still being young and impressionable, he moved on to Montana, the state he calls home. There he continued to enjoy the solitude of nature, and discovered not only its beauty, but the source of all beauty. He also developed a voracious appetite for books, which opened his mind to a vast world of ideas. In high-school, beginning to see past appearances to the deeper beauty and greater wonder that is in the order of things, he solemnly decided that he would become an astrophysicist, even though he had trouble pronouncing this word signifying his future career title. Nevertheless, his calling was affirmed when he took his first physics course, which sadly didn't happen until his senior year of high-school. Matriculating with honors, he eagerly entered the world of higher education at Whitman College, in Walla Walla, Washington in 1983. There he was not disappointed, but found the intellectual challenge and education he sought, and there also learned the value and necessity of human friendships. After four very formative years he graduated with a Bachelor of Arts in physics and astronomy, and went on the next year to the University of Florida for graduate studies. This work represents the culmination of his time there.


I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.


James H. Hunter, Jr.
Professor of Astronomy and
Adjunct Professor of Physics


I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.


Stanley Dermott, Chair and
Professor of Astronomy

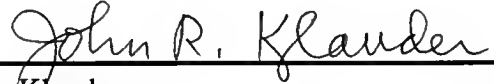
I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.


Haywood Smith
Associate Professor of Astronomy

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.


Henry E. Kandrup
Associate Professor of Astronomy

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

A handwritten signature in cursive script that reads "John R. Klauder". The signature is written in black ink and is positioned above a horizontal line.

John Klauder
Professor of Physics

This dissertation was submitted to the Graduate Faculty of the Department of Astronomy in the College of Liberal Arts and Sciences, and to the Graduate School, and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August 1995

Dean, Graduate School

LD
1780
1995
.D779

